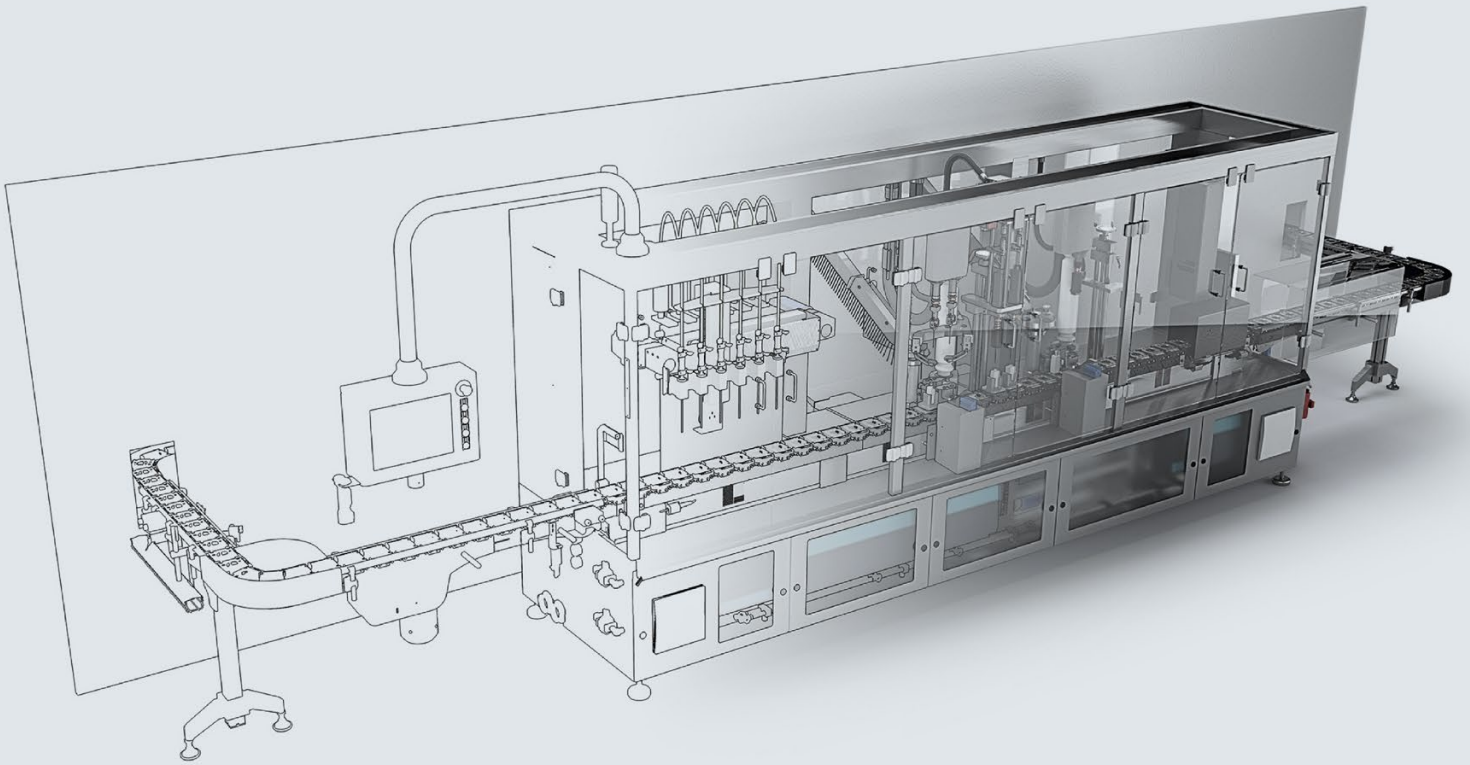
SIEMENS



Funktionshandbuch

SIMATIC

S7-1500

S7-PLCSIM Advanced

Ausgabe 11/2019

support.industry.siemens.com

SIEMENS

SIMATIC

S7-1500
S7-PLCSIM Advanced

Funktionshandbuch

11/2019
A5E37039506-AC

Rechtliche Hinweise

Warnhinweiskonzept

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

⚠GEFAHR
bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠WARNUNG
bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠VORSICHT
bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

ACHTUNG
bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung **qualifiziertem Personal** gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

Bestimmungsgemäßer Gebrauch von Siemens-Produkten

Beachten Sie Folgendes:

⚠WARNUNG
Siemens-Produkte dürfen nur für die im Katalog und in der zugehörigen technischen Dokumentation vorgesehenen Einsatzfälle verwendet werden. Falls Fremdprodukte und -komponenten zum Einsatz kommen, müssen diese von Siemens empfohlen bzw. zugelassen sein. Der einwandfreie und sichere Betrieb der Produkte setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung, Montage, Installation, Inbetriebnahme, Bedienung und Instandhaltung voraus. Die zulässigen Umgebungsbedingungen müssen eingehalten werden. Hinweise in den zugehörigen Dokumentationen müssen beachtet werden.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
DEUTSCHLAND

A5E37039506-AC
℗ 09/2019 Änderungen vorbehalten

Vorwort

Zweck der Dokumentation

Das vorliegende Funktionshandbuch beschreibt die Simulations-Software SIMATIC S7-PLCSIM Advanced V3.0. Sie können damit Ihre SIMATIC STEP 7-Programme auf einem virtuellen Controller simulieren und testen.

Gültigkeitsbereich

Das Funktionshandbuch ist gültig für folgende Bestellvarianten:

- 6ES7823-1FA01-0YA5 - SIMATIC S7-PLCSIM Advanced V3.0 Floating License (DVD)
- 6ES7823-1FE01-0YA5 - SIMATIC S7-PLCSIM Advanced V3.0 Floating License (Download)
- 6ES7823-1FA01-0YE5 - Upgrade SIMATIC S7-PLCSIM Advanced V2.0 → V3.0 (DVD)
- 6ES7823-1FE01-0YE5 - Upgrade SIMATIC S7-PLCSIM Advanced V2.0 → V3.0 (Download)

Die Artikel beinhalten jeweils eine Lizenz für zwei Instanzen.

Erforderliche Grundkenntnisse

Die Software darf nur von qualifiziertem Personal verwendet werden.
Vorausgesetzt werden folgende Kenntnisse:

- Industrieautomatisierung und Automatisierungstechnik
- Programmierung mit STEP 7 (TIA Portal)
- SIMATIC CPUs und CPU-Programmierung
- PC-basierte Automatisierung mit S7-1500 und mit WinCC Runtime Advanced
- Kenntnisse der Programmierung mit C++ oder C#
- PC-Technik
- Betriebssystem Windows

Konventionen

Konventionen STEP 7: Zur Bezeichnung der Projektier- und Programmiersoftware verwenden wir in der vorliegenden Dokumentation "STEP 7" als Synonym für alle Versionen von "STEP 7 (TIA Portal)".

Für SIMATIC S7-PLCSIM Advanced V3.0 verwenden wir auch kurz "PLCSIM Advanced".

Beachten Sie auch die folgendermaßen gekennzeichneten Hinweise:

Hinweis

Ein Hinweis enthält wichtige Informationen zum in der Dokumentation beschriebenen Produkt, zur Handhabung des Produkts oder zu dem Teil der Dokumentation, auf den besonders aufmerksam gemacht werden soll.

Besondere Informationen

Hinweis

Liesmich

Aktualisierungen zum Funktionshandbuch erhalten Sie als Download im Internet (https://support.industry.siemens.com/cs/de/de/view/109739154).

Anwendungsbeispiele

Folgende Anwendungsbeispiele zu S7-PLCSIM Advanced finden Sie im Internet:

- SIMATIC S7-PLCSIM Advanced: Co-Simulation via API (1 (https://support.industry.siemens.com/cs/ww/de/view/109739660))
- Digitalisierung mit TIA Portal: Virtuelle Inbetriebnahme mit SIMATIC und Simulink (2 (https://support.industry.siemens.com/cs/document/109749187))

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 einen Bestandteil eines solchen Konzepts.

Die Kunden sind dafür verantwortlich, unbefugten Zugriff auf ihre Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Diese Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und nur wenn entsprechende Schutzmaßnahmen (z. B. Firewalls und/oder Netzwerksegmentierung) ergriffen wurden.

Weiterführende Informationen zu möglichen Schutzmaßnahmen im Bereich 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, Produkt-Updates anzuwenden, sobald sie 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

Tabellen

Bilder

Wegweiser

1

1.1 Wegweiser Dokumentation

Die Dokumentation für das Automatisierungssystem SIMATIC S7-1500 und das Dezentrale Peripheriesystem SIMATIC ET 200SP gliedert sich in drei Bereiche.

Basisinformationen

Systemhandbücher und Getting Started beschreiben ausführlich die Projektierung, Montage, Verdrahtung und Inbetriebnahme der Systeme SIMATIC S7-1500 und ET 200SP. Die Online-Hilfe von STEP 7 unterstützt Sie bei der Projektierung und Programmierung.

Geräteinformationen

Gerätehandbücher enthalten eine kompakte Beschreibung der modulspezifischen Informationen wie Eigenschaften, Anschlussbilder, Kennlinien, Technische Daten.

Übergreifende Informationen

In den Funktionshandbüchern finden Sie ausführliche Beschreibungen zu übergreifenden Themen, z. B. Diagnose, Kommunikation, Motion Control, Webserver, OPC UA.

Die Dokumentation finden Sie zum kostenlosen Download im Internet (https://support.industry.siemens.com/cs/ww/de/view/109742691).

Änderungen und Ergänzungen zu den Handbüchern werden in Produktinformationen dokumentiert.

Sie finden die Produktinformationen im Internet:

- S7-1500 (https://support.industry.siemens.com/cs/de/de/view/68052815)
- ET 200SP (https://support.industry.siemens.com/cs/de/de/view/73021864)

Manual Collections

Die Manual Collections beinhalten die vollständige Dokumentation zu den Systemen zusammengefasst in einer Datei.

Sie finden die Manual Collections im Internet:

- S7-1500 (https://support.industry.siemens.com/cs/ww/de/view/86140384)
- ET 200SP (https://support.industry.siemens.com/cs/ww/de/view/84133942)

"mySupport"

Mit "mySupport", Ihrem persönlichen Arbeitsbereich, machen Sie das Beste aus Ihrem Industry Online Support.

In "mySupport" können Sie Filter, Favoriten und Tags ablegen, CAx-Daten anfordern und sich im Bereich Dokumentation Ihre persönliche Bibliothek zusammenstellen. Des Weiteren sind in Support-Anfragen Ihre Daten bereits vorausgefüllt und Sie können sich jederzeit einen Überblick über Ihre laufenden Anfragen verschaffen.

Um die volle Funktionalität von "mySupport" zu nutzen, müssen Sie sich einmalig registrieren.

Sie finden "mySupport" im Internet (http://support.industry.siemens.com/My/ww/de/documentation).

"mySupport" - Dokumentation

In "mySupport" haben Sie im Bereich Dokumentation die Möglichkeit ganze Handbücher oder nur Teile daraus zu Ihrem eigenen Handbuch zu kombinieren.
Sie können das Handbuch als PDF-Datei oder in einem nachbearbeitbaren Format exportieren.

Sie finden "mySupport" - Dokumentation im Internet (https://support.industry.siemens.com/My/ww/de/).

Anwendungsbeispiele

Die Anwendungsbeispiele unterstützen Sie mit verschiedenen Tools und Beispielen bei der Lösung Ihrer Automatisierungsaufgaben. Dabei werden Lösungen im Zusammenspiel mehrerer Komponenten im System dargestellt - losgelöst von der Fokussierung auf einzelne Produkte.

Sie finden die Anwendungsbeispiele im Internet (https://support.industry.siemens.com/sc/ww/de/sc/2054).

TIA Selection Tool

Mit dem TIA Selection Tool können Sie Geräte für Totally Integrated Automation (TIA) auswählen, konfigurieren und bestellen. Es fasst die bereits bekannten Konfiguratoren für die Automatisierungstechnik in einem Werkzeug zusammen.

Mit dem TIA Selection Tool erzeugen Sie aus Ihrer Produktauswahl oder Produktkonfiguration eine vollständige Bestellliste.

Sie finden das TIA Selection Tool im Internet (http://w3.siemens.com/mcms/topics/de/simatic/tia-selection-tool).

SIMATIC Automation Tool

Mit dem SIMATIC Automation Tool können Sie unabhängig vom TIA Portal gleichzeitig an verschiedenen SIMATIC S7-Stationen Inbetriebsetzungs- und Servicetätigkeiten als Massenoperation ausführen.

Das SIMATIC Automation Tool bietet eine Vielzahl von Funktionen:

- Scannen eines PROFINET/Ethernet Anlagennetzes und Identifikation aller verbundenen CPUs
- Adresszuweisung (IP, Subnetz, Gateway) und Stationsname (PROFINET Device) zu einer CPU
- Übertragung des Datums und der auf UTC-Zeit umgerechneten PG/PC-Zeit auf die Baugruppe
- Programm-Download auf CPU
- Betriebsartenumstellung RUN/STOP
- CPU-Lokalisierung mittels LED-Blinken
- Auslesen von CPU-Fehlerinformation
- Lesen des CPU Diagnosepuffers
- Rücksetzen auf Werkseinstellungen
- Firmwareaktualisierung der CPU und angeschlossener Module

Sie finden das SIMATIC Automation Tool im Internet (https://support.industry.siemens.com/cs/ww/de/view/98161300).

PRONETA

Mit SIEMENS PRONETA (PROFINET Netzwerk-Analyse) analysieren Sie im Rahmen der Inbetriebnahme das Anlagennetz. PRONETA verfügt über zwei Kernfunktionen:

- Die Topologie-Übersicht scannt selbsttätig das PROFINET und alle angeschlossenen Komponenten.
- Der IO-Check ist ein schneller Test der Verdrahtung und des Modulausbaus einer Anlage.

Sie finden SIEMENS PRONETA im Internet (https://support.industry.siemens.com/cs/ww/de/view/67460624).

SINETPLAN

SINETPLAN, der Siemens Network Planner, unterstützt Sie als Planer von Automatisierungsanlagen und -netzwerken auf Basis von PROFINET. Das Tool erleichtert Ihnen bereits in der Planungsphase die professionelle und vorausschauende Dimensionierung Ihrer PROFINET-Installation. Weiterhin unterstützt Sie SINETPLAN bei der Netzwerkoptimierung und hilft Ihnen, Netzwerkressourcen bestmöglich auszuschöpfen und Reserven einzuplanen. So vermeiden Sie Probleme bei der Inbetriebnahme oder Ausfälle im Produktivbetrieb schon im Vorfeld eines geplanten Einsatzes. Dies erhöht die Verfügbarkeit der Produktion und trägt zur Verbesserung der Betriebssicherheit bei.

Die Vorteile auf einen Blick

- Netzwerkoptimierung durch portgranulare Berechnung der Netzwerklast
- höhere Produktionsverfügbarkeit durch Onlinescan und Verifizierung bestehender Anlagen
- Transparenz vor Inbetriebnahme durch Import und Simulierung vorhandener STEP7 Projekte
- Effizienz durch langfristige Sicherung vorhandener Investitionen und optimale Ausschöpfung der Ressourcen

Sie finden SINETPLAN im Internet (https://www.siemens.com/sinetplan).

1.2 S7-PLCSIM Produkte

PLCSIM Advanced V3.0, PLCSIM V16 und PLCSIM V5.x

Tabelle 1- 1 Vergleich von S7-PLCSIM Produkten

Funktion	PLCSIM Advanced V3.0	PLCSIM V16	PLCSIM V5.x
Runtime	Eigenständig	Zusammen mit STEP 7	Zusammen mit STEP 7
Bedienoberfläche	Control Panel	Look&Feel von TIA Portal	Look&Feel von STEP 7 V5.x
Kommunikation	Softbus, TCP/IP	Nur Softbus	Nur Softbus
Unterstützte CPU-Familien	S7-1500 (C, T, F), ET 200SP, ET 200SP F	S7-1200 (F), S7-1500 (C, T, F), ET 200SP, ET 200SP F	S7-300, S7-300F S7-400, S7-400F
API für Co-Simulation[1]	✓	-	-
Webserver	✓, nur über TCP/IP	-	-
ODK	✓	-	-
OPC UA	✓, nur über TCP/IP	-	-
Prozessdiagnose	✓	✓	-
S7-Kommunikation	✓	Über Softbus	Über Softbus
Open User-Kommunikation	✓, UDP nur über TCP/IP	Über Softbus	-
Traces[2]	✓	(✓)	-
Motion[3]	✓	(✓)	-
Geschützte Bausteine (KHP)	✓	✓, nur für S7-1500 CPUs	-
Multiple Instanzen	Bis zu 16	Bis zu 2	-
Verteilte Instanzen	✓, nur über TCP/IP	-	-
Virtuelle Zeit	✓	-	-
Anschluss realer CPUs/HMIs	✓, nur über TCP/IP	-	-
DNS Nutzung	✓	-	-
Virtuelle Memory Card	✓	-	-
Kommunikation zwischen den Instanzen	-	PLCSIM ab V12 und PLCSIM V5.x können auf dem selben PC oder der selben virtuellen Maschine installiert und betrieben werden.	
	-	Instanzen von PLCSIM ab V12 können über Softbus mit PLCSIM V5.x kommunizieren.	
	PLCSIM Advanced V3.0 und PLCSIM ab V15 können auf dem selben PC oder der selben virtuellen Maschine installiert und betrieben werden. Die Kommunikation zwischen beiden Anwendungen ist **nicht** simulierbar.		-
	PLCSIM V5.4 SP8 wird automatisch mit PLCSIM Advanced installiert. Die Kommunikation zwischen beiden Anwendungen ist simulierbar. Instanzen von PLCSIM Advanced können über Softbus mit PLCSIM V5.4 SP8 kommunizieren.		

1 Über C++ und C#-Programme und Simulations-Software

2 Bei PLCSIM V16 im TIA Portal beobachtbar; bei PLCSIM Advanced V3.0 zusätzlich auch im Webserver beobachtbar.

3 Bei PLCSIM V16 werden die Achsen immer unabhängig von der Achskonfiguration simuliert betrieben.
Bei PLCSIM Advanced V3.0 können die Achsen über die API im "Real"-Modus betrieben werden.

Produktübersicht

2

2.1 Was ist S7-PLCSIM Advanced?

Mit PLCSIM Advanced können Sie Ihre CPU-Programme auf einem virtuellen Controller simulieren. Sie benötigen dafür keine realen Controller. Sie können Ihre CPU mit STEP 7 im TIA Portal konfigurieren, Ihre Anwendungslogik programmieren und dann die Hardware-Konfiguration und das Programm in den virtuellen Controller laden. Von dort können Sie Ihre Programmlogik ausführen, die Auswirkungen der simulierten Eingänge und Ausgänge beobachten und Ihre Programme anpassen.

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

PLCSIM Advanced ermöglicht über die **Anwenderschnittstelle (API)** die Interaktion mit eigenen C++/C#-Programmen oder mit Simulations-Software.

Anwendungsgebiete

- Verifikation des Anwenderprogramms (TIA Portal)
- Automatisiertes Testen des STEP 7-Programms
- Software in the Loop-Simulation zur virtuellen Inbetriebnahme von Werkzeug-/Produktionsmaschinen, Produktionszellen und Produktionslinien in einer Anlage

Vorteile

Der Einsatz von PLCSIM Advanced bietet zahlreiche Vorteile:

- Qualität der Automatisierungsprojekte verbessern durch frühzeitige Fehlererkennung
- Kosten für Hardware in Simulationsumgebungen vermeiden
- Verkürzung der Inbetriebnahmezeit
- Risiken bei der Inbetriebnahme reduzieren
- Frühes Training der Bediener möglich
- Effizienz der Fertigung durch Optimierung von Programmteilen steigern
- Effizienz beim Tausch von Maschinen-Komponenten steigern
- Effizienz bei der Erweiterung existierender Anlagen steigern

2.2 Kompatibilität beim Upgrade

Kompatibilität von API- und Runtime-Versionen

PLCSIM Advanced V3.0 enthält die Runtime-Version V3.0 und die API-Versionen V1.0 (SP1) bis V3.0.

Die Installation von PLCSIM Advance V3.0 führt zu einem Upgrade einer vorhandenen früheren Version. Der Runtime Manager von PLCSIM Advanced V3.0 ist kompatibel mit Projekten, die mit früheren API-Version erstellt wurden. Bereits erstellte Projekte können Sie daher weiter verwenden.

Hinweis

Eine API mit einer höheren Versionsnummer (z. B. V3.0) kann sich nicht mit einer früheren Runtime-Version (z. B. V1.0) verbinden.

Kompatibilität zum TIA Portal und zu CPU Firmware-Versionen

Die in PLCSIM Advanced V3.0 verwendete Firmware entspricht der einer CPU S7-15xx V2.8.

Die Firmware ist kompatibel zu den TIA Portal Versionen V14 bis V16.

Tabelle 2- 1 Kompatibilität zu CPU Firmware-Versionen

PLCSIM Advanced	Unterstützte CPU Firmware-Version
V1.0 SP1	V1.8, V2.0
V2.0	V1.8 bis V2.5
V2.0 SP1	V1.8 bis V2.6
V3.0	V1.8 bis V2.8

2.3 Sicherheit bei S7-PLCSIM Advanced

Einschränkungen bei der Sicherheit

Beachten Sie beim Einsatz von PLCSIM Advanced folgende Einschränkungen:

Authentifizierung

- Die Anwenderschnittstellen (API) verfügen nicht über Authentifizierungs- und Autorisierungsmöglichkeiten. Es gibt keinen Schutz über Anwenderprofile und Passworte.
- Die Runtime Manager Kommunikation ist nicht über eine Authentifizierung geschützt.

Kommunikation

- Die rechnerübergreifende Simulations-Kommunikation ist nicht verschlüsselt.
- Bei netzwerkübergreifender Kommunikation wird auf dem PC ein TCP/IP Port geöffnet.
- Die mitinstallierte Programmbibliothek WinPcap ermöglicht den Zugriff auf die TCP/IP Netzwerk-Kommunikation.

Hinweis

Bei rechnerübergreifender Kommunikation wird empfohlen, ein abgeschlossenes Simulations-Netzwerk zu nutzen, das nicht mit einen Produktiv-Netzwerk verbunden ist.

Know-how-Schutz

Hinweis

Know-how-geschützte Bausteine

Wenn know-how-geschützte Bausteine für den Simulations-Support freigeschaltet werden, ist der Know-how-Schutz eingeschränkt.

Hinweis

CPU Funktionsbibliotheken bei ODK

Die SO-Dateien bei ODK sind nicht know-how-geschützt. Die Verantwortung für die SO-Dateien und deren Know-How-Schutz liegt beim Kunden.

2.4 Simulations-Support

Voraussetzung für Simulationen

Hinweis

Simulierbarkeit aktivieren

Um ein STEP 7-Projekt mit der Simulation zu nutzen, müssen Sie in den Eigenschaften des Projekts im Register "Schutz" die Option "Beim Übersetzen von Bausteinen Simulierbarkeit unterstützen" aktivieren und mit OK bestätigen.

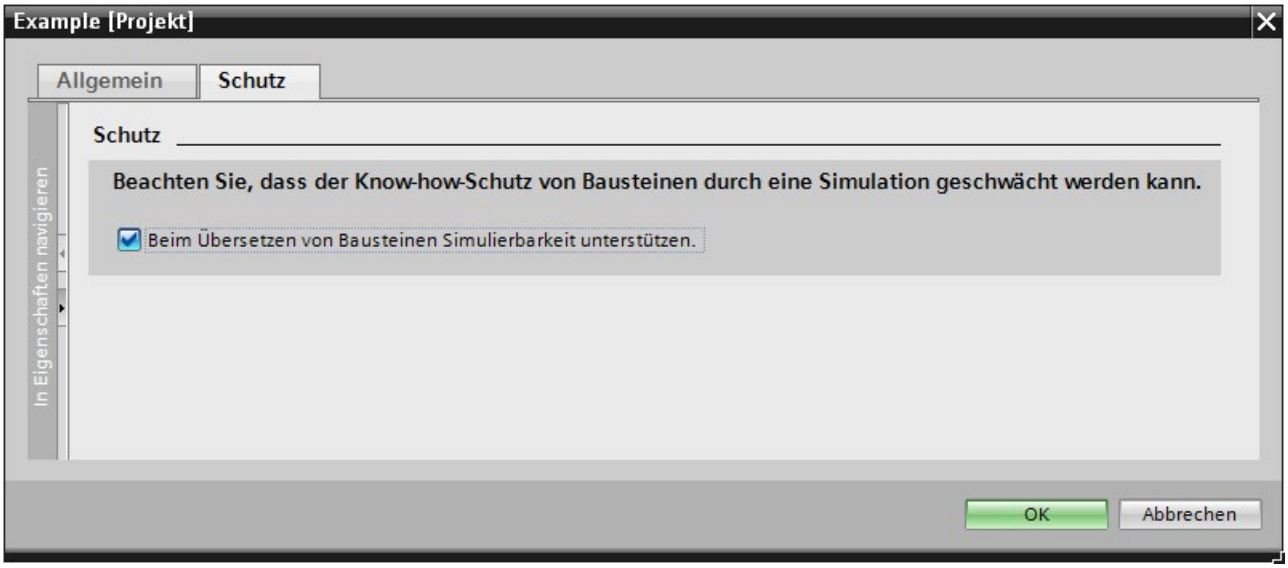


Bild 2-1 Simulierbarkeit aktivieren

Know-how-Schutz

Wenn ein know-how-geschützter Baustein für die Simulation verwendet werden soll, muss er durch Passworteingabe entsperrt werden. Erst durch die Entsperrung kann in den Eigenschaften des Bausteins im Register "Allgemein > Übersetzung" die Option "Simulierbarkeit mit SIMATIC S7-PLCSIM (Advanced)" aktiviert werden. Weitere Informationen erhalten Sie im Internet (https://support.industry.siemens.com/cs/ww/de/view/109754928).

Globale Bibliotheken

Bei Verwendung von globalen Bibliotheken kann der Know-how-Schutz nicht gesetzt werden, da die Bibliotheken schreibgeschützt sind.

Die Option "Simulierbarkeit mit SIMATIC S7-PLCSIM (Advanced)" muss beim Erzeugen der Bausteine gesetzt werden (Quelle der Bausteine).

2.5 Unterstützte CPUs

Unterstützte CPUs aus der S7-1500-Familie

PLCSIM Advanced V3.0 unterstützt die Simulation folgender CPUs:

Tabelle 2- 2 Unterstützte CPUs

Typ	Version V1.8 bis V2.8	
Standard-CPUs / Fehlersichere CPUs	CPU 1511-1 PN	CPU 1511F-1 PN
	CPU 1513-1 PN	CPU 1513F-1 PN
	CPU 1515-2 PN	CPU 1515F-2 PN
	CPU 1516-3 PN/DP	CPU 1516F-3 PN/DP
	CPU 1517-3 PN/DP	CPU 1517F-3 PN/DP
	CPU 1518-4 PN/DP	CPU 1518F-4 PN/DP
	CPU 1518-4 PN/DP ODK	CPU 1518F-4 PN/DP ODK
	CPU 1518-4 PN/DP MFP	CPU 1518F-4 PN/DP MFP
Kompakt-CPUs[1]	CPU 1511C-1 PN	
	CPU 1512C-1 PN	
ET 200SP-CPUs	CPU 1510SP-1 PN	CPU 1510SP F-1 PN
	CPU 1512SP-1 PN	CPU 1512SP F-1 PN
Technologie-CPUs	CPU 1511T-1 PN	CPU 1511TF-1 PN
	CPU 1515T-2 PN	CPU 1515TF-2 PN
	CPU 1516T-3 PN/DP	CPU 1516TF-3 PN/DP
	CPU 1517T-3 PN/DP	CPU 1517TF-3 PN/DP

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

Nicht unterstützte CPUs

PLCSIM Advanced unterstützt die Simulation folgender CPUs nicht:

- S7-1500R/H-CPUs
- S7-1200-CPUs
- ET 200pro, ET 200pro F CPUs
- ET 200SP Open Controller CPU 1515SP PC
- Software Controller

2.6 Unterschiede zwischen simulierter und realer CPU

Der virtuelle Controller kann eine reale CPU nicht vollständig bis in die einzelnen Details simulieren. Auch wenn ein Programm fehlerfrei auf die CPU heruntergeladen wird und erfolgreich läuft, bedeutet dies nicht unbedingt, dass sich der virtuelle Controller bei der Simulation genauso verhält wie eine reale CPU.

Deterministik

PLCSIM Advanced wird auf einem PC mit dem Betriebssystem Windows ausgeführt. Daher sind die Scanzykluszeit und die genaue Zeit von Aktionen in PLCSIM Advanced nicht die gleichen, wie wenn diese Aktionen auf physischer Hardware ausgeführt würden. Dies liegt daran, dass sich auf Ihrem PC mehrere Programme die Verarbeitungsressourcen teilen.

Um unter diesen Voraussetzungen die bestmögliche Deterministik zu bieten, benötigt PLCSIM Advanced ab V2.0 einen freien Core (CPU-Kern) pro Instanz.

Wenn Ihr Programm stark von der Zeit abhängt, die für die Ausführung von Aktionen benötigt wird, dann berücksichtigen Sie, dass Sie Ihr Programm nicht nur auf Basis der Zeitergebnisse der Simulation beurteilen.

Know-how-Schutz

Projekte mit Know-how-Schutz für Bausteine können nur simuliert werden, wenn diese für die Simulation freigeschaltet sind. Hierfür benötigen Sie das Passwort des Bausteins.

Anweisungen

Anweisungen werden bis auf wenige Ausnahmen simuliert, siehe Einschränkungen bei Anweisungen (Seite 400).

Programme, die sich auf die entsprechenden Anweisungen stützen, verhalten sich bei der Simulation anders als bei realen CPUs.

Anzeige des Mengengerüsts

In STEP 7 wird in der Projektnavigation unter "Program info" für alle CPUs das maximale Mengengerüst angezeigt, das auf der CPU 1518-4 PN/DP basiert.

Unter "Online & diagnositics" wird das maximale Mengengerüst der aktuell simulierten CPU angezeigt.

2.6.1 Einschränkungen bei allen unterstützten CPUs

Bussysteme

PLCSIM Advanced simuliert keine Bussysteme (PROFINET IO, PROFIBUS DP, Rückwandbus).

Peripherie

PLCSIM Advanced simuliert die reale CPU, nicht aber projektierte Peripheriemodule und die Onboard-Peripherie der Kompakt-CPUs.

Kommunikationsmodule

PLCSIM Advanced unterstützt keine Kommunikationsmodule und die damit verbundenen Features wie "Zugriff auf PLC über Kommunikationsmodul".

Teilprozessabbilder

Teilprozessabbilder werden wie bei der realen CPU unterstützt.

Addressbereiche, die nicht einem Teilprozessabbild zugeordnet sind, werden am Zykluskontrollpunkt aktualisiert.

Diagnose / Diagnosemeldungen

Mit PLCSIM Advanced können einfache Diagnosepuffer-Einträge nach PROFINET-Standard simuliert werden.

PROFIBUS-spezifische Diagnosen (z. B. über DS0, DS1) und anwenderspezifische Textlisten werden nicht unterstützt.

Online- und Diagnosefunktionen

Bestimmte Online- und Diagnosefunktionen haben in der Simulation keinen sinnvollen Zweck und werden daher nicht unterstützt. Dazu gehören z. B. die Funktionen "Speicherkarte formatieren" und "Firmware-Update".

Status-Anzeigen LED-Blinken

In STEP 7 können Sie über das Dialogfeld "Erweitertes Laden in Gerät" die LED-Anzeigen an einer CPU blinken lassen. PLCSIM Advanced simuliert diese Funktion nicht.

Datenprotokollierung

PLCSIM Advanced simuliert die Datenprotokollierung nicht.

Rezepte

PLCSIM Advanced simuliert die Verwendung von Rezepten nicht.

Kopierschutz

PLCSIM Advanced simuliert nicht Kopierschutz.

Eingeschränkte Unterstützung

PLCSIM Advanced simuliert einige Funktionen in eingeschränktem Umfang. Eine Übersicht finden Sie im Kapitel Einschränkungen, Meldungen und Abhilfe (Seite 394).

2.6.2 Hinweise

Passwort-Übernahme bei Baugruppentausch (S7-1500)

Je nach Firmware-Version der betroffenen CPUs (zu tauschende CPU und Austausch-CPU) werden Sie mit Informationen konfrontiert, die Ihnen ein Update auf einen aktuellen Algorithmus anbieten bzw. Sie dazu auffordern, neue Passwörter zu vergeben, da die Austausch-CPU die vorhandene Passwort-Projekierung nicht übernehmen kann.

Wenn sich die zu tauschende CPU und die Austausch-CPU hinsichtlich des verwendeten Algorithmus identisch verhalten, ist keine Aktion erforderlich; die Passwort-Projektierung wird wie die übrigen Parametereinstellungen übernommen.

PLCSIM Advanced unterstützt keine Passwortverschlüsselung für CPU-Versionen mit Firmware kleiner als V2.0.

Um in der Simulation Schutzstufen, den Webserver und den Zugriffsschutz der F-CPU nutzen zu können, klicken Sie auf die Schaltfläche "Passwortverschlüsselung aktualisieren". Die Schaltfläche finden Sie in den CPU-Eigenschaften im Register "Schutz & Security" unter "Zugriffsstufe".

HMI-Geräte und CPU Schutzstufen

- PLCSIM Advanced unterstützt HMI-Geräte ab Version 14. Verbindungen zu HMI-Geräten vor V14 werden nicht unterstützt.
- PLCSIM Advanced unterstützt Schutzstufen, wenn der virtuelle S7-1500 Controller mit einer Firmware-Version V2.0 oder höher projektiert ist.
- Verbindungen von HMI-Geräten ab V14 zu virtuellen S7-1500 Controllern, die mit einer Firmware-Version V2.0 oder höher projektiert sind, sind mit oder ohne Schutzstufen möglich.
- Verbindungen von HMI-Geräten ab V14 zu virtuellen S7-1500 Controllern, die mit einer Firmware-Version kleiner als V2.0 projektiert sind, sind ohne Schutzstufen möglich.

Abhilfe

Um eine Verbindung zum HMI-Gerät V13 oder früher herzustellen, müssen Sie ein Update dieses HMI-Geräts auf Stand V14 durchführen.

Um eine Verbindung vom virtuellen Controller, der mit einer Firmware Version kleiner als V2.0 projektiert ist, zum HMI-Gerät herzustellen, müssen Sie vorhandene Schutzstufen aus dem Projekt entfernen.

Safety-System-Version V1.6 bzw. V2.0 für fehlersichere Peripherie

Um ein Projekt mit fehlersicheren Eingabe- und Ausgabemodulen erfolgreich zu simulieren und zu testen, müssen Sie für das Projekt die Safety-System-Version V1.6 bzw. V2.0 verwenden. Mit einer älteren Version funktioniert die Simulation der fehlersicheren Eingabe- und Ausgabemodule nicht korrekt.

Priorität beim Prozessalarm-OB

Die Prozessalarme, die über die PLCSIM Advanced API ausgelöst werden, werden sequenziell in das Anwenderprogramm übertragen.

Nur wenn Ereignisse gleichzeitig eintreffen, entscheidet die Priorität des zugeordneten Prozessalarm-OBs über die Reihenfolge der Ausführung.

Installieren

3

3.1 Einleitung

3.1.1 Systemanforderungen

PLCSIM Advanced installieren Sie bevorzugt auf einem SIMATIC Field PG M5 Advanced oder einem vergleichbaren PC.

Damit PLCSIM Advanced effizient funktioniert, müssen folgende Mindestvoraussetzungen bei der Computer-Hardware oder einer Virtuellen Maschine gegeben sein.

Tabelle 3- 1 Systemanforderungen

	Hardware	Virtuelle Maschine
Prozessor	• Ein logischer Intel Core™ i7-6820EQ Kern pro gestarteter Instanz • Mindestens ein weiterer Kern für das Betriebssystem • Mindestens ein weiterer Kern für zusätzliche aktive Anwendungen	• Eine virtuelle CPU pro gestarteter Instanz muss der VM zugewiesen sein • Eine entsprechende Anzahl Prozessoren muss physikalisch auf dem Host existieren • Mindestens ein weiterer Kern für das Betriebssystem • Mindestens ein weiterer Kern für zusätzliche aktive Anwendungen • Mindestens zwei Kerne, wenn STEP 7 (TIA Portal) auf der VM installiert wird
RAM	• 1 GB pro gestarteter Instanz • Mindestens 4 GB für das Windows Betriebssystem • Zusätzlicher RAM entsprechend den Anforderungen der übrigen aktiven Anwendungen	• 1 GB pro gestarteter Instanz • Mindestens 4 GB für das Windows Betriebssystem • Zusätzlicher RAM entsprechend den Anforderungen der übrigen aktiven Anwendungen • Mindestens 8 GB, wenn STEP 7 (TIA Portal) auf der VM installiert wird
Freier Festplattenspeicher	5 GB	5 GB
Bildschirmauflösung	Mindestens 1024 x 768	Mindestens 1024 x 768

Betriebssysteme (64 Bit-Varianten)

PLCSIM Advanced V3.0 unterstützt folgende Betriebssysteme:

- Windows 7 Home Premium
- Windows 7 Professional SP1
- Windows 7 Enterprise SP1
- Windows 7 Ultimate SP1
- Windows 10 Home Version 1809, 1903
- Windows 10 Pro Version 1809, 1903
- Windows 10 Enterprise Version 1809, 1903 (für SIMATIC Field PG M5)
- Windows 10 (IoT) Enterprise 2016 LTSB
- Windows 10 (IoT) Enterprise 2019 LTSC
- Windows Server 2012 R2 StdE (vollständige Installation)
- Windows Server 2016 Standard (vollständige Installation)
- Windows Server 2019 Standard (vollständige Installation)

Hinweis

Stellen Sie sicher, dass das eingesetzte Windows Betriebssystem auf dem aktuellsten Stand ist.

Virtualisierungsplattformen

Sie können STEP 7 und PLCSIM Advanced in einer virtuellen Maschine installieren. Verwenden Sie zu diesem Zweck eine der folgenden Virtualisierungsplattformen in der angegebenen oder einer neueren Version:

- VMware vSphere Hypervisor (ESXi) 6.7
- VMware Workstation Pro 15.0.2
- VMware Player 15.0.2

Informationen, die Sie beachten müssen, wenn Sie STEP 7 (TIA Portal) auf einer virtuellen Maschine installieren, finden Sie im Internet (https://support.industry.siemens.com/cs/ww/de/view/78788417).

3.1.2 Einschränkungen durch Virenscanner

ACHTUNG
Einschränkungen durch Virenscanner Virenscanner, die das Verhalten von Prozessen und die Kommunikation überwachen, können erheblichen Einfluss auf die Performance der Laufzeit und der Kommunikation von PLCSIM Advanced haben.

Hinweis

Liesmich

Aktualisierungen zum Thema erhalten Sie als Download im Internet (https://support.industry.siemens.com/cs/de/de/view/109739154.).

Unterstützte Virenscanner

PLCSIM Advanced unterstützt Trend Micro Office Scan 12.0.

Bekannte Probleme und Einschränkungen

Kaspersky

Beim Einsatz des Virenscanners Anti-Virus von Kaspersky kann es vorkommen, dass während der Installation von PLCSIM Advanced die Netzwerkeinstellungen nicht korrekt gesetzt werden. Dies führt dazu, dass die Kommunikation über TCP/IP nicht genutzt werden kann (Fehlercode -50).

Abhilfe

Kontrollieren Sie Ihre Netzwerkeinstellungen wie im Kapitel Verteilte Kommunikation aktivieren (Seite 59) beschrieben.

3.1.3 Lizenzen

Floating License

PLCSIM Advanced wird mit einer Lizenz vom Typ Floating ausgeliefert. Diese kann lokal abgelegt und für ein Netzwerk freigegeben werden.

Hinweis
Gültigkeit

Eine Lizenz ist gültig für zwei Instanzen innerhalb einer PLCSIM Advanced Installation.

PLCSIM Advanced V3.0 kann nur mit einer V3.0 Lizenz genutzt werden.

Den Umgang mit Lizenzen finden Sie in der Hilfe zum SIMATIC Automation License Manager (ALM).

3.1.4 Trial License

Für S7-PLCSIM Advanced V3.0 steht eine Lizenz zur Verfügung, die auf 21 Tage begrenzt ist. Nach Ablauf diser Trial License wird die Instanz nicht mehr gestartet.

Trial License aktivieren

Sobald Sie im Control Panel eine Instanz starten, sucht der Automation License Manager (ALM) im Netzwerk nach einer gültigen Lizenz. Wenn für S7-PLCSIM Advanced keine Floating License vorliegt, bietet der ALM die Trial License zur Aktivierung an.

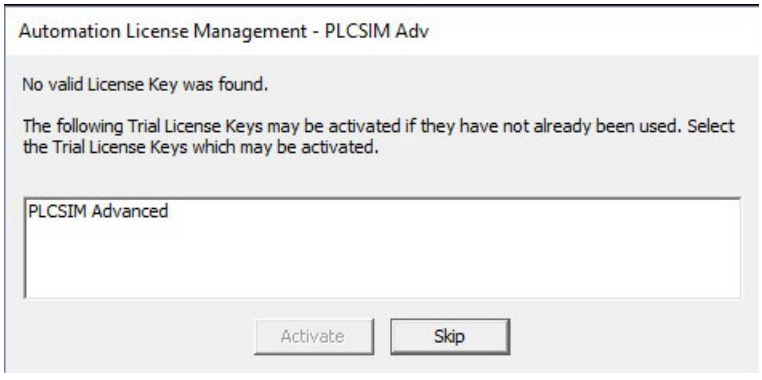


Bild 3-1 Trial License aktivieren

Eine Meldung zeigt beim Starten der Instanzen die noch verbleibenden Tage an.

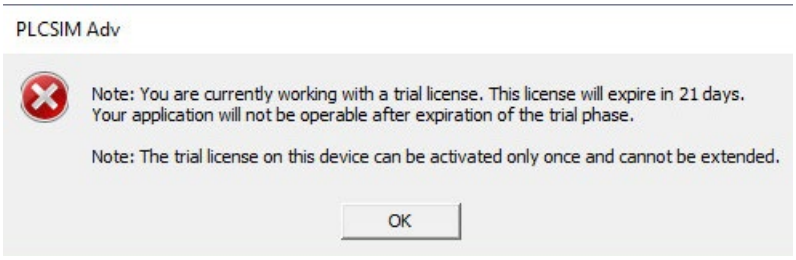


Bild 3-2 Trial License Meldung

Hinweis

Remote-Zugriff

Bei einem Remote-Zugriff muss die Meldung auf dem PC bestätigt werden, auf dem die Instanz gestartet wurde.

Timeout Meldung

Wenn die Meldung zur Lizenz nicht rechtzeitig bestätigt wird, wird die Instanz nicht gestartet und folgende Meldung erscheint:

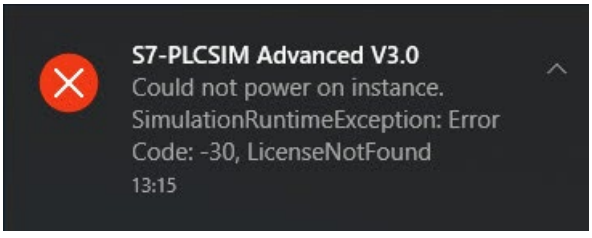


Bild 3-3 Timeout Meldung

Abhilfe

Starten Sie die Instanz erneut und bestätigen Sie die Meldung zur Lizenz.

API-Funktionen zu Lizenzen

PLCSIM Advanced überprüft regelmäßig, ob eine Lizenz vorhanden ist. Folgende Rückgabewerte geben dabei Aufschluss über den Status (beispielhaft für C++):

- Rückgabewerte für die API-Funktion `PowerOn()` und die Callback-Funktion `OnOperatingStateChanged`
  - `SREC_OK`, wenn eine Floating License vorliegt.
  - `SREC_WARNING_TRIAL_MODE_ACTIVE`, wenn eine Instanz mit der Trial License gestartet ist.
  - `SREC_WARNING_RUNNING_ON_TIA_PORTAL_TEST_SUITE`, wenn keine gültige Lizenz für PLCSIM Advanced vorhanden ist, jedoch eine "TIA Portal Test Suite" Lizenz. PLCSIM Advanced startet mit dieser Lizenz. Ein Download aus dem TIA Portal ist möglich, allerdings beendet sich die Instanz ohne Rückmeldung, wenn der Download nicht aus der TIA Portal Test Suite erfolgt ist.
  - `SREC_NOT_EMPTY`, wenn keine gültige Lizenz für PLCSIM Advanced vorhanden ist, jedoch eine "TIA Portal Test Suite" Lizenz. In diesem Fall wird der Hochlauf aus der Virtual SIMATIC Memory Card nicht unterstützt.
- Rückgabewert für die Callback-Funktion `OnOperatingStateChanged`
  - `SREC_LICENSE_NOT_FOUND`, wenn die Instanz nach Ablauf der 21 Tage abgeschaltet wird.

3.1.5 Installationsprotokoll

In Protokolldateien werden automatisch Informationen über die folgenden Installationsprozesse aufgezeichnet:

- Installation von S7-PLCSIM Advanced
- Änderung oder Aktualisierung der Installation von S7-PLCSIM Advanced
- Reparatur einer vorhandenen Installation von S7-PLCSIM Advanced
- Deinstallation von S7-PLCSIM Advanced

Mit den Protokolldateien können Sie Installationsfehler und Warnungen auswerten. Die Fehlerbehebung der Installation können Sie selbst durchführen oder sich an den technischen Support von Siemens wenden. Der Produkt-Support benötigt die Informationen aus dem Installationsprotokoll zur Analyse des Problems. Senden Sie dazu den Ordner mit den Protokolldateien als ZIP-Datei an den Support.

Speicherort des Installationsprotokolls

Der Speicherort der Protokolldatei richtet sich nach dem Betriebssystem. Geben Sie in die Adresszeile im Windows-Explorer die Umgebungsvariable "%autinstlog%" ein, um den Ordner mit den Protokolldateien zu öffnen. Alternativ gelangen Sie zum entsprechenden Verzeichnis, indem Sie in der Befehlszeile "cd %autinstlog%" eingeben.

Die Protokolldateien heißen:

- "SIA_S7-PLCSIM_Advanced_V03@<DATUM_UHRZEIT>.log"
- "SIA_S7-PLCSIM_Advanced_V03@<DATUM_UHRZEIT>_summary.log"

Setup_Report (CAB-Datei)

Das Installationsprotokoll und weitere erforderliche Dateien werden in einer Archivdatei gespeichert. Diese finden Sie unter "%autinstlog%\Reports\Setup_report.cab".

Für jede Installation wird eine eigene CAB-Datei mit einer Datums-ID gespeichert.

Wenn Sie Hilfe bei der Installation benötigen, dann senden Sie dem technischen Support von Siemens diese CAB-Datei zur Fehlerbehebung.

3.2 S7-PLCSIM Advanced

Das S7-PLCSIM Advanced Paket enthält folgende Software:

- S7-PLCSIM Advanced
- Automation License Manager
- S7-PLCSIM V5.4
- .NET Framework
- WinPcap

Das Paket steht als Download und auf DVD zur Verfügung:

- SIMATIC S7-PLCSIM Advanced V3.0 Floating License
- Upgrade SIMATIC S7-PLCSIM Advanced V2.0 → V3.0

Bewahren Sie die DVD nach der Installation an einem sicheren, leicht zugänglichen Ort auf.

Setup-Programm

Mit dem Setup-Programm können Sie Ihre Installation bei Bedarf ändern, reparieren oder deinstallieren.

3.3 S7-PLCSIM Advanced installieren

Voraussetzung für die Installation

Das Setup-Programm startet automatisch mit einem Doppelklick auf das Download-Paket oder wenn Sie die DVD in das Laufwerk einlegen. Vergewissern Sie sich, dass die folgenden Bedingungen erfüllt sind, bevor Sie mit dem Installationsvorgang beginnen:

- Hardware und Software des Computers erfüllen die Systemvoraussetzungen.
- Sie haben Administratorrechte auf dem Installationscomputer.
- Keine anderen Programme sind aktiv. Dies gilt auch für den Siemens Automation License Manager und andere Anwendungen von Siemens.
- Alle S7-PLCSIM Versionen kleiner oder gleich V14 sind deinstalliert.

Hinweis

Sicherheitseinstellungen

Für die Lizenzierung über den ALM müssen Sie bei der Installation zustimmen, dass der Port 4410 für TCP als Ausnahme in der Windows-Firewall eingegeben wird (Schritt 5).

Hinweis

Einsatz von Virenscannern

Beachten Sie die Hinweise im Kapitel Einschränkungen durch Virenscanner (Seite 43).

S7-PLCSIM Advanced installieren

Gehen Sie zur Installation wie folgt vor:

1. Doppelklicken Sie auf das Download-Paket oder legen Sie das Installationsmedium in das DVD-Laufwerk Ihres Computers ein. Das Setup-Programm startet automatisch, sofern Sie nicht die Autostartfunktion auf dem Computer deaktiviert haben. Wenn das Setup-Programm nicht automatisch startet, starten Sie es manuell durch Doppelklick auf die Datei "Start.exe". Das Fenster "General settings" wird angezeigt.
2. Klicken Sie auf die Schaltfläche "Installationshinweise lesen". Wenn Sie die Hinweise gelesen haben, schließen Sie die Datei.
3. Klicken Sie auf die Schaltfläche "Produktinformationen lesen". Wenn Sie die Informationen gelesen haben, schließen Sie die Datei.
4. Klicken Sie auf die Schaltfläche "Durchsuchen", wenn Sie den Standardinstallationspfad ändern möchten. Der Installationspfad darf maximal 89 Zeichen lang sein. Der Pfadname darf keine UNICODE-Zeichen enthalten. Wenn Sie nicht den Standardinstallationspfad auswählen, wird das Desktop-Symbol möglicherweise nicht richtig angezeigt.
5. Klicken Sie auf die Schaltfläche "Weiter". Das Fenster mit den Sicherheitseinstellungen wird angezeigt. Zur Fortsetzung der Installation aktivieren Sie das Optionskästchen am unteren Rand des Bildschirms, um Änderungen an den Sicherheits- und Berechtigungseinstellungen Ihres Systems zu akzeptieren.
6. Klicken Sie auf die Schaltfläche "Weiter". Das Fenster mit den Installationseinstellungen wird angezeigt. Sie können einen Bericht der Einstellungen speichern oder drucken, indem Sie auf "Bericht speichern" oder "Bericht drucken" klicken. Überprüfen Sie die Einstellungen auf ihre Richtigkeit. Wenn Sie Änderungen vornehmen möchten, klicken Sie auf die Schaltfläche "Zurück", bis Sie an die Stelle im Installationsprozess kommen, an der Sie die Änderungen vornehmen können. Nachdem Sie Ihre Änderungen abgeschlossen haben, klicken Sie auf "Weiter".
7. Der Übersichtsbildschirm zeigt Ihre Installationsdetails an. Klicken Sie auf die Schaltfläche "Installieren". Daraufhin wird die Installation gestartet.
8. Nach dem Abschluss des Setup-Programms müssen Sie Ihren Computer neu starten. Wählen Sie "Ja, Computer jetzt neu starten", um den Computer sofort neu zu starten oder wählen Sie "Nein, Computer später starten", um den Neustart später durchzuführen.
9. Klicken Sie auf "Neu starten". Wenn der Computer nicht neu gestartet wird, klicken Sie auf "Beenden".

Fehler beim Installieren von S7-PLCSIM Advanced

Beim Installieren wird eine vorhandene Installation von S7-PLCSIM angezeigt.

Eine Voraussetzung für die Installation von S7-PLCSIM Advanced ist, dass sich keine S7-PLCSIM Installation kleiner oder gleich V14 auf dem gleichen Computer befindet.

Obwohl in der Liste "Programme und Funktionen" keine Installation von S7-PLCSIM angezeigt wird, kann diese noch auf dem Computer vorhanden sein.

Abhilfe

Führen Sie für S7-PLCSIM kleiner oder gleich V14 das Setup durch und deinstallieren Sie dabei das Programm.

Wenn das Setup nicht mehr vorhanden ist, laden Sie die Setup-Dateien von S7-PLCSIM über die Siemens Mall (https://support.industry.siemens.com/cs/document/65601780).

3.4 S7-PLCSIM Advanced ändern

Voraussetzungen

Folgende Bedingungen müssen erfüllt sein, bevor Sie mit dem Ändern der Installation beginnen:

- Hardware und Software des Computers erfüllen die Systemvoraussetzungen.
- Sie haben Administratorrechte auf dem Installationscomputer.
- Keine anderen Programme sind aktiv.

Vorgehensweise

Gehen Sie zum Ändern Ihrer S7-PLCSIM Advanced-Installation wie folgt vor:

1. Doppelklicken Sie auf das Download-Paket oder legen Sie das Installationsmedium in das Laufwerk ein. Das Setup-Programm startet automatisch, sofern Sie nicht die Autostartfunktion auf dem Computer deaktiviert haben. Wenn das Setup-Programm nicht automatisch startet, starten Sie es manuell durch Doppelklick auf die Datei "Start.exe".
2. Befolgen Sie die Eingabeaufforderungen, bis Sie in das Fenster "Konfiguration" gelangen.
3. Wählen Sie das Optionskästchen "Upgrade ändern".
4. Befolgen Sie die weiteren Eingabeaufforderungen, um Ihre Installation zu ändern.
5. Um den Installationsvorgang abzuschließen, starten Sie Ihren Computer neu.

Hinweis
Zielverzeichnis

Das Zielverzeichnis können Sie nicht ändern, weil Sie eine bestehende Installation ändern.

3.5 S7-PLCSIM Advanced reparieren

Voraussetzungen

Folgende Bedingungen müssen erfüllt sein, bevor Sie mit dem Reparieren der Installation beginnen:

- Hardware und Software erfüllen die Systemvoraussetzungen.
- Sie haben Administratorrechte auf dem Installationscomputer.
- Keine anderen Programme sind aktiv.

Vorgehensweise

Gehen Sie zum Reparieren Ihrer Installation wie folgt vor:

1. Doppelklicken Sie auf das Download-Paket oder legen Sie das Installationsmedium in das Laufwerk ein. Das Setup-Programm startet automatisch, sofern Sie nicht die Autostartfunktion auf dem Computer deaktiviert haben. Wenn das Setup-Programm nicht automatisch startet, starten Sie es manuell durch Doppelklick auf die Datei "Start.exe".
2. Befolgen Sie die Eingabeaufforderungen, bis Sie in das Fenster "Konfiguration" gelangen. Wählen Sie das Optionskästchen "Reparieren".
3. Befolgen Sie die weiteren Eingabeaufforderungen, um Ihre Installation zu reparieren.
4. Um den Reparaturvorgang abzuschließen, starten Sie Ihren Computer neu.

3.6 S7-PLCSIM Advanced deinstallieren

Sie haben zwei Möglichkeiten, S7-PLCSIM Advanced zu deinstallieren:

- Sie deinstallieren über die Windows-Systemsteuerung das Programm.
- Sie deinstallieren über das Setup-Programm das gesamte Produkt.

S7-PLCSIM Advanced über die Windows-Systemsteuerung deinstallieren

Gehen Sie wie folgt vor:

1. Wählen Sie in der Windows-Systemsteuerung die Option "Programme und Funktionen".
2. Klicken Sie mit der rechten Maustaste auf "Siemens S7-PLCSIM Advanced V3.0" und wählen Sie "Deinstallieren".
3. Befolgen Sie zur Deinstallation die Eingabeaufforderungen.
4. Um den Deinstallationsvorgang abzuschließen, starten Sie Ihren Computer neu.

   Wenn Sie keinen Neustart durchführen, läuft der Runtime Manager weiter.

Wenn beim Deinstallieren über die Windows-Systemsteuerung Probleme auftreten, verwenden Sie zum Deinstallieren der Anwendung das Installationsmedium, sofern vorhanden.

S7-PLCSIM Advanced über das Setup-Programm deinstallieren

Gehen Sie wie folgt vor:

1. Doppelklicken Sie auf das Download-Paket oder legen Sie das Installationsmedium in das Laufwerk ein. Das Setup-Programm startet automatisch, sofern Sie nicht die Autostartfunktion auf dem Computer deaktiviert haben. Wenn das Setup-Programm nicht automatisch startet, starten Sie es manuell durch Doppelklick auf die Datei "Start.exe".

   Wenn Sie keinen Neustart durchführen, läuft der Runtime Manager weiter.

2. Befolgen Sie die Eingabeaufforderungen, bis Sie in das Fenster "Konfiguration" gelangen. Ihre vorherige Installation wird erkannt. Wählen Sie das Optionskästchen "Deinstallieren".
3. Befolgen Sie zur Deinstallation die weiteren Eingabeaufforderungen.
4. Um den Deinstallationsvorgang abzuschließen, starten Sie Ihren Computer neu.

   Wenn Sie keinen Neustart durchführen, läuft der Runtime Manager weiter.

Weitere Software deinstallieren

Beim Deinstallieren bleibt folgende Software aus dem S7-PLCSIM Advanced Paket installiert:

- Automation License Manager
- S7-PLCSIM V5.4
- .NET Framework
- WinPcap

Wenn Sie auch diese Software deinstallieren möchten, dann nutzen Sie dazu die Windows-Systemsteuerung.

Kommunikationswege

4

Lokale und verteilte Kommunikation

Für die Kommunikation zwischen STEP 7 ab V15 und den Instanzen der PLCSIM Advanced Anwenderschnittstellen stehen folgende Wege offen:

Tabelle 4- 1 Lokale und verteilte Kommunikation

Kommunikationswege	Lokal	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
Kommunikation...			
zwischen STEP 7 und Instanzen	Ja	Ja	Ja
zwischen Instanzen untereinander	Ja	Ja	Ja
über OPC UA Server und Webserver	Nein	Ja	Ja
zwischen einer Instanz und einer realen Hardware-CPU	Nein	Nein	Ja
zwischen einer Instanz und einer realen HMI ab V14	Nein	Nein	Ja
zwischen einer Instanz und einer simulierten HMI ab V14	Ja	Ja	Ja

Softbus

Softbus ist ein Kommunikationsweg über eine virtuelle Software-Schnittstelle.

Die Kommunikation ist beschränkt auf einen lokalen PC oder eine virtuelle Maschine. Der Vorteil hierbei ist, dass keine Daten versehentlich auf eine Hardware-CPU heruntergeladen werden können oder mit realer Hardware kommuniziert wird.

Kommunikationsschnittstelle wählen

Die Kommunikationsschnittstelle programmieren Sie über die Anwenderschnittstellen (API) oder wählen Sie im Control Panel unter "Online Access". Die Einstellung gilt jeweils für alle erzeugten Instanzen. Die Voreinstellung ist die Kommunikation über "PLCSIM" (Softbus).

Für die verteilte Kommunikation über den "PLCSIM Virtual Ethernet Adapter" (TCP/IP) sind weitere Netzwerkeinstellungen erforderlich, siehe Netzwerk-Adressen in der Simulation (Seite 72).

API-Funktionen zur Auswahl der Kommunikationsschnittstelle

- GetCommunicationInterface() (Seite 154)
- SetCommunicationInterface() (Seite 154)
- CommunicationInterface { get; set; } (Seite 155)

Siehe auch

Schnittstellen - Informationen und Einstellungen (Seite 151)

4.1 Lokale Kommunikation

Die lokale Kommunikation kann über die Protokolle Softbus oder über TCP/IP erfolgen.

Bei der lokalen Kommunikation befindet sich die PLCSIM Advanced Instanz auf dem selben PC oder auf der selben Virtualisierungsplattform wie STEP 7 oder ein anderer Kommunikations-Partner.

Lokale Kommunikation über Softbus

In PLCSIM Advanced ist die lokale Kommunikation über Softbus voreingestellt.

Dadurch ist sichergestellt, dass keine Daten versehentlich auf eine Hardware-CPU heruntergeladen werden oder dass mit realer Hardware kommuniziert wird.

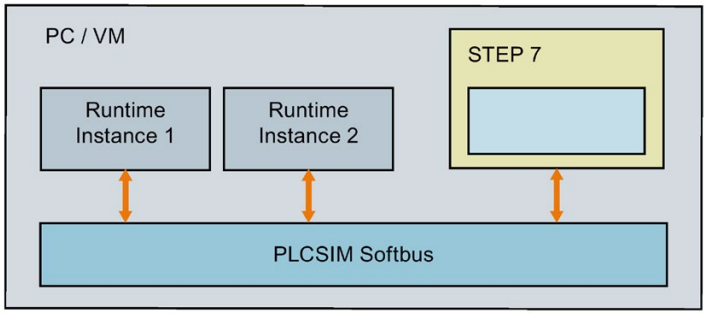


Bild 4-1 Lokale Kommunikation über Softbus

Lokale Kommunikation über TCP/IP

Die Kommunikation erfolgt über den PLCSIM Virtual Ethernet Adapter, eine virtuelle Netzwerk-Schnittstelle, die sich wie eine reale Netzwerk-Schnittstelle verhält.

Hinweis

Lokale Kommunikation über TCP/IP

Stellen Sie sicher, dass die Kommunikation nur lokal erfolgt und kein Download auf reale Hardware durchgeführt werden kann. Dazu darf im physikalischen Netzwerk und im Subnetz-Protokoll des PLCSIM Virtual Ethernet Adapters kein weiterer Adapter Ihres Windows-PC konfiguriert sein. Hintergrund ist der Microsoft KB 175767.

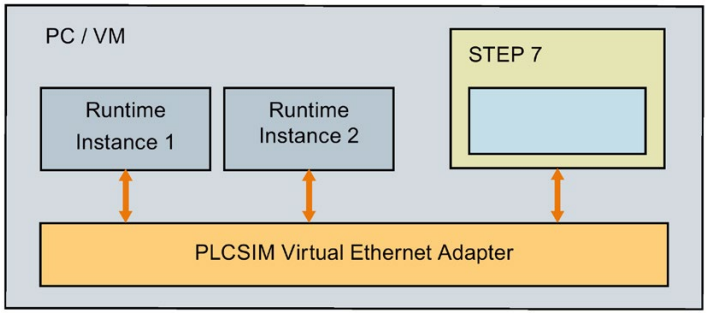


Bild 4-2 Lokale Kommunikation über TCP/IP

Weitere Informationen

Siehe Fehlercode SREC_COMMUNICATION_INTERFACE_NOT_AVAILABLE für die Funktion `PowerOn()` im Kapitel Betriebszustand (Seite 168).

4.2 Kommunikation über TCP/IP

Verteilte Kommunikation

Verteilte Kommunikation über TCP/IP bedeutet, dass die PLCSIM Advanced Instanzen über den Virtual Switch mit anderen Geräten kommunizieren. Kommunikation ist möglich mit realen oder simulierten CPUs, realen oder simulierten HMIs.

Der PLCSIM Virtual Switch muss am PLCSIM Virtual Ethernet Adapter aktiviert werden, damit die Instanzen im Netzwerk sichtbar sind.

Beispiel 1: Verteilte Kommunikation

Im folgenden Beispiel befindet sich STEP 7 auf einem PC, die PLCSIM Advanced Instanzen auf einem weiteren PC oder einer virtuellen Maschine. Die PCs sind über ihre realen Ethernet Adapter verbunden.

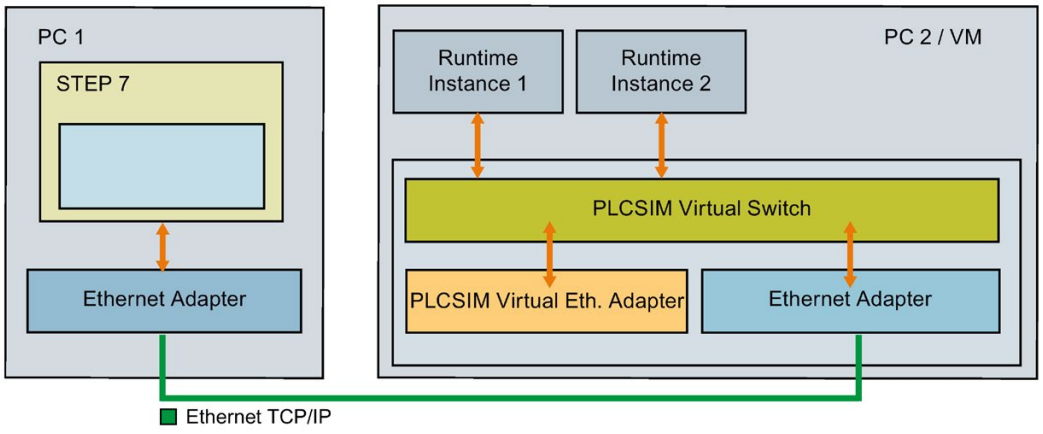


Bild 4-3 Verteilte Kommunikation über Ethernet

Beispiel 2: Verteilte Kommunikation auf einem PC

Im folgenden Beispiel befindet sich STEP 7 auf einem PC, die PLCSIM Advanced Instanzen in einer virtuellen Maschine auf dem selben PC. PC und virtuelle Maschine sind über die (virtuellen) Netzwerkkarten verbunden.

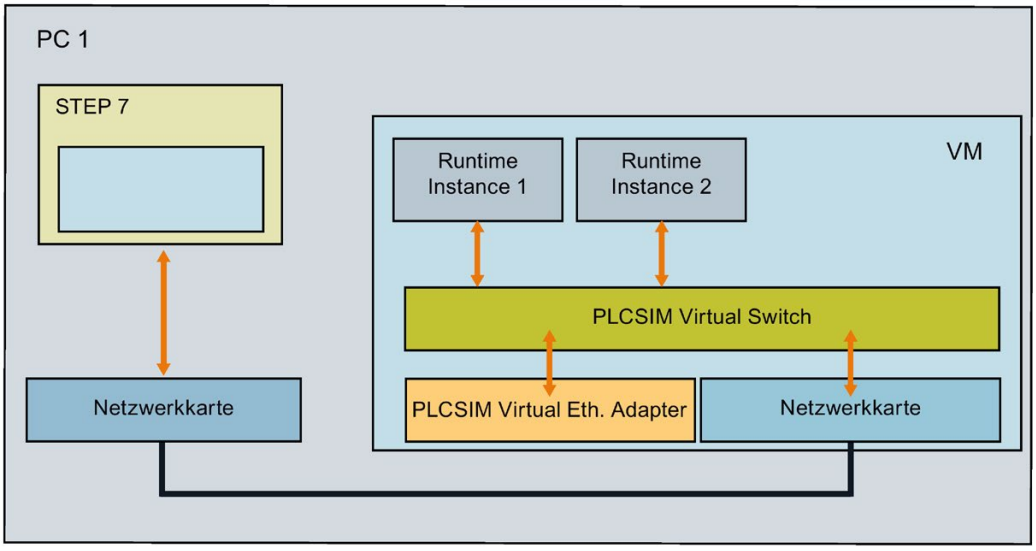


Bild 4-4 Verteilte Kommunikation über Netzwerkkarten

Notwendige Einstellungen im Dialog "Virtual Machine Settings"

Wenn Sie innerhalb der virtuellen Maschine STEP 7 (TIA Portal) und Ihr Projekt geöffnet haben, aktivieren Sie für Ihre Onlineverbindung die folgenden Optionen:

1. Öffnen Sie den Dialog "Virtual Machine Settings" über den Menübefehl "Player > Manage > Virtual Machine Settings".
2. Klicken Sie anschließend im Register "Hardware" auf "Network Adapter" und aktivieren Sie im rechten Fenster die folgenden Optionen:
   - Connected
   - Connect at power on
   - Bridged: Connected directly to the physical network
   - Replicate physical network connection state
3. Klicken Sie auf die Schaltfläche "Configure Adapters" und aktivieren Sie Ihre Netzwerkverbindung, z. B. "Intel(R)82574L LM Gigabit Network Connection".
4. Bestätigen Sie die Einstellung mit OK und beenden Sie den Dialog "Virtual Machine Settings" mit OK.

Beispiel 3: Verteilte Kommunikation

Das folgende Beispiel zeigt einen Aufbau mit PCs, auf denen verteilt STEP 7, PLCSIM Advanced Instanzen und virtuelle Maschinen mit PLCSIM Advanced Instanzen laufen.

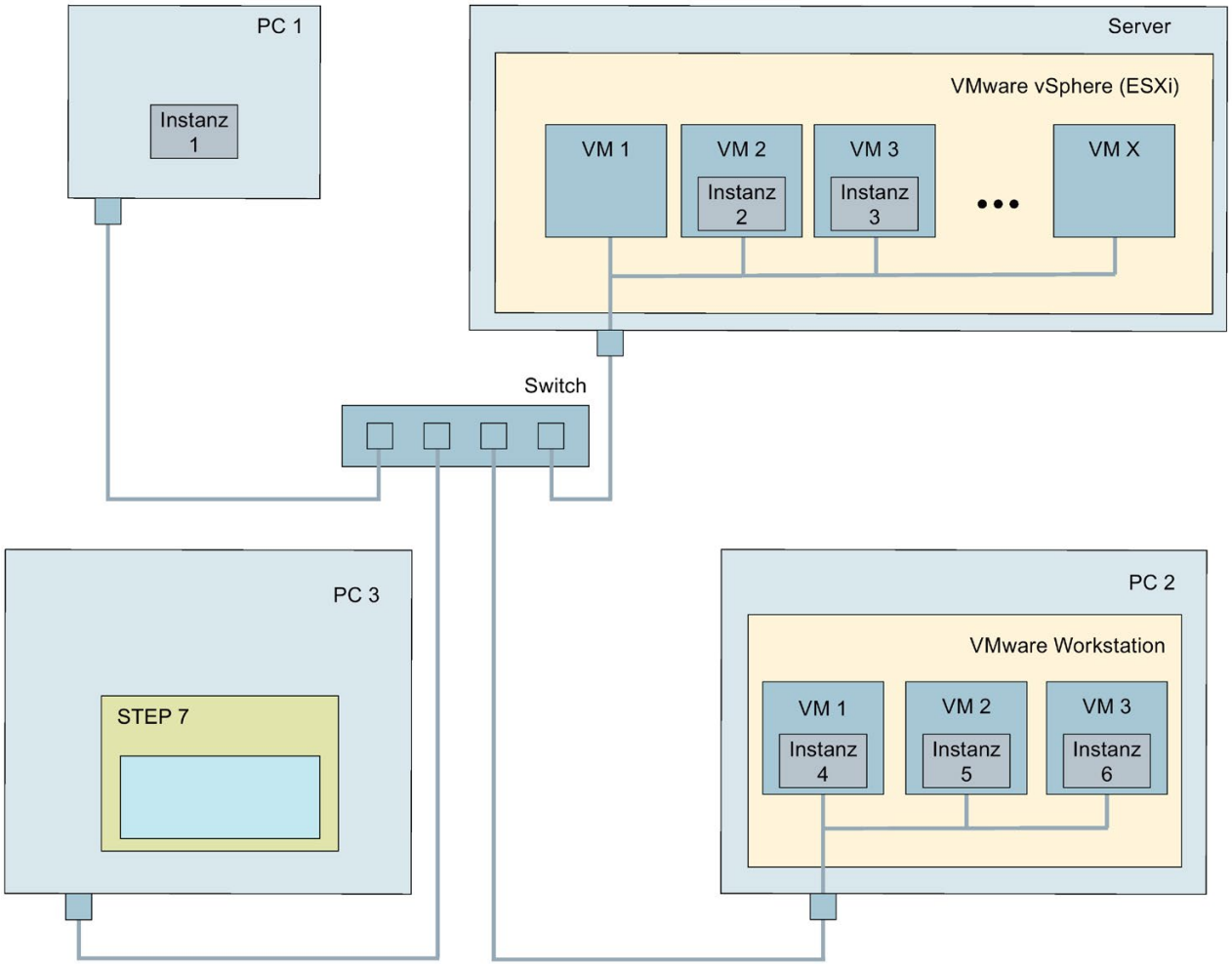


Bild 4-5 Verteilte Kommunikation mit PCs und virtuellen Maschinen

4.3 Verteilte Kommunikation aktivieren

Über die Voreinstellung kann der PLCSIM Virtual Switch nur lokal kommunizieren. Damit eine verteilte, d. h. rechnerübergreifende Kommunikation möglich ist, müssen Sie bei einer realen Netzwerkkarte den PLCSIM Virtual Switch aktivieren.

Hinweis

Netzwerkkarte

Stellen Sie sicher, dass nur bei einer einzigen Netzwerkkarte der PLCSIM Virtual Switch aktiviert ist. Das Control Panel von PLCSIM Advanced prüft die Aktivierung und meldet ggf. eine fehlerhafte Konfiguration (Fehlercode -50).

PLCSIM Virtual Switch aktivieren

Um die PLCSIM Instanzen im Netzwerk sichtbar zu machen und um andere Teilnehmer zu erreichen, aktivieren Sie den PLCSIM Virtual Switch im Control Panel von PLCSIM Advanced oder unter Windows:

1. Öffnen Sie dazu in der Windows Systemsteuerung das "Netzwerk- und Freigabecenter".
2. Öffnen Sie die Eigenschaften des gewünschten Netzwerkadapters, z. B. der "Local Area Connection".
3. Aktivieren Sie das Optionskästchen für den "Siemens PLCSIM Virtual Switch" und bestätigen Sie mit OK.

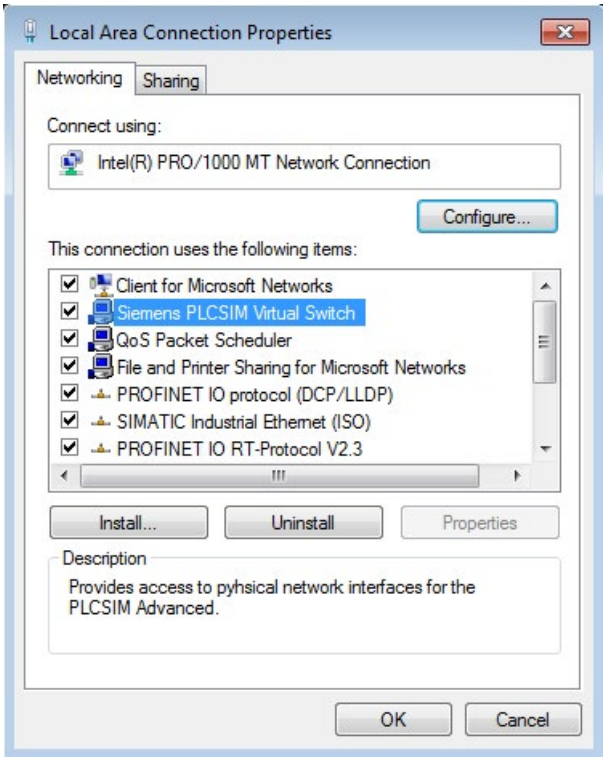


Bild 4-6 PLCSIM Virtual Switch aktivieren

Erreichbare Teilnehmer

Wenn der PLCSIM Virtual Switch aktiviert ist, zeigt STEP 7 in der Projektnavigation die Teilnehmer an, die über den Virtual Ethernet Adapter erreichbar sind.

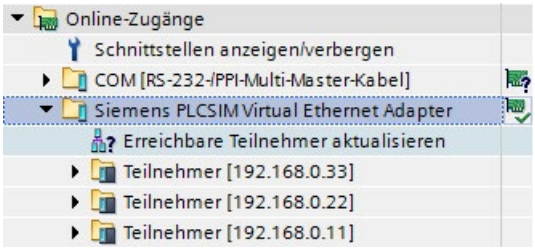


Bild 4-7 Erreichbare Teilnehmer am Virtual Ethernet Adapter

Verteilte Kommunikation über WLAN

Bei der Verwendung von verteilter Kommunikation über WLAN kann es sein, dass die von PLCSIM Advanced installierte Programmbibliothek WinPcap mit dem integrierten WLAN-Adapter des Computers nicht funktioniert. In diesem Fall kann keine WLAN-Verbindung aufgebaut werden.

Abhilfe

Verwenden Sie den kabelgebundenen Netzwerkadapter des PCs/Notebooks und schalten Sie einen WLAN-Adapter vor.

Simulation 5

5.1 CPU simulieren

5.1.1 Prinzipielles Vorgehen bei der Simulation

Die folgende Übersicht zeigt die prinzipiellen Schritte, um eine Simulation mit einer Instanz eines virtuellen Controllers durchzuführen.

Voraussetzung

Folgende Voraussetzungen müssen zum Starten einer Simulation über die lokale Kommunikation erfüllt sein:

- STEP 7 ab V14 und S7-PLCSIM Advanced V3.0 sind auf dem selben PC installiert.
- In STEP 7 ist die Hardware-CPU konfiguriert.

Hinweis

Simulations-Support aktivieren

Aktivieren Sie in STEP 7 in den Eigenschaften des Projekts im Register "Schutz" das Optionskästchen "Beim Übersetzen von Bausteinen Simulierbarkeit unterstützen"; siehe Simulations-Support (Seite 35).

Über das Control Panel Instanz erstellen und einschalten

- PLCSIM Advanced Control Panel öffnen (siehe Kapitel Control Panel - Bedienoberfläche (Seite 62))
- Optionen "Start Virtual S7-1500 PLC" aufklappen
- Namen für eine Instanz eingeben
- CPU-Typ auswählen
- Instanz erstellen über die Schaltfläche "Start"

In STEP 7 Download durchführen und Simulation starten

- Programm Download auf den virtuellen Controller durchführen (siehe Kapitel Download (Seite 70))
- Virtuellen Controller in RUN schalten, um Simulation zu starten
- Diagnose durchführen

5.1.2 Control Panel - Bedienoberfläche

5.1.2.1 S7-PLCSIM Advanced Symbol

Nach der Installation von PLCSIM Advanced befindet sich folgendes Symbol auf dem Windows Desktop:

Bild 5-1 PLCSIM Advanced Symbol

Ein Doppelklick auf das Symbol öffnet das Control Panel für PLCSIM Advanced. Wenn sich das Control Panel im Hintergrund befindet, wird es durch einen weiteren Doppelklick in den Vordergrund gebracht.

Über Windows-Funktionen können Sie das Symbol im Infobereich der Taskleiste dauerhaft einblenden.

Grafische Oberfläche öffnen

Ein Rechtsklick auf das Symbol in der Taskleiste startet das Control Panel als Schnellansicht. Ein Doppelklick startet das Control Panel als Fenster.

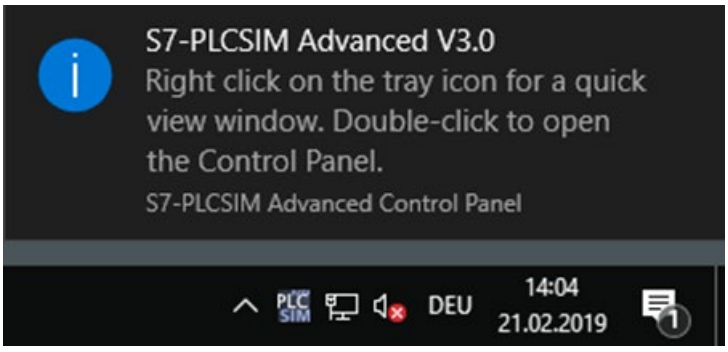


Bild 5-2 Grafische Oberfläche öffnen

In der Taskleiste können Sie die Mouse-over-Funktion nutzen, um Meldungen zum aktuellen Zustand der Instanzen anzuzeigen.

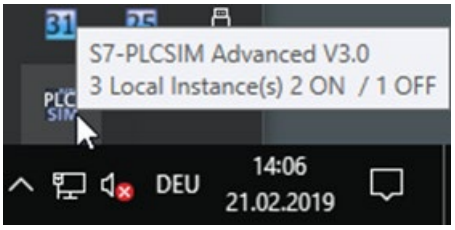


Bild 5-3 Beispiel: Meldung in der Taskleiste

5.1.2.2 Grafische Oberflächen

Die grafischen Oberflächen synchronisieren sich mit API-Befehlen. Sie sind aber optional und werden zum Betrieb von PLCSIM Advanced über die API nicht benötigt.

S7-PLCSIM Advanced V3.0 stellt das Control Panel mit zwei Ansichten zur Verfügung.

- **Control Panel als Schnellansicht**

  Ein Rechtsklick auf das Symbol in der Taskleiste startet die Schnellansicht.

  Ein Klick auf eine freie Fläche auf dem Desktop minimiert die Schnellansicht. Die Instanzen bleiben davon unberührt.

- **Control Panel als Fenster**

  Ein Doppelklick auf das Symbol auf dem Desktop oder in der Taskleiste startet das Control Panel als Fenster.

Control Panel als Fenster

Im Gegensatz zur Schnellansicht können Sie das Control Panel über Schaltflächen in der Titelleiste bedienen.

Sie können dieses Fenster schließen, ohne dass der Simulation Runtime Prozess beendet wird.

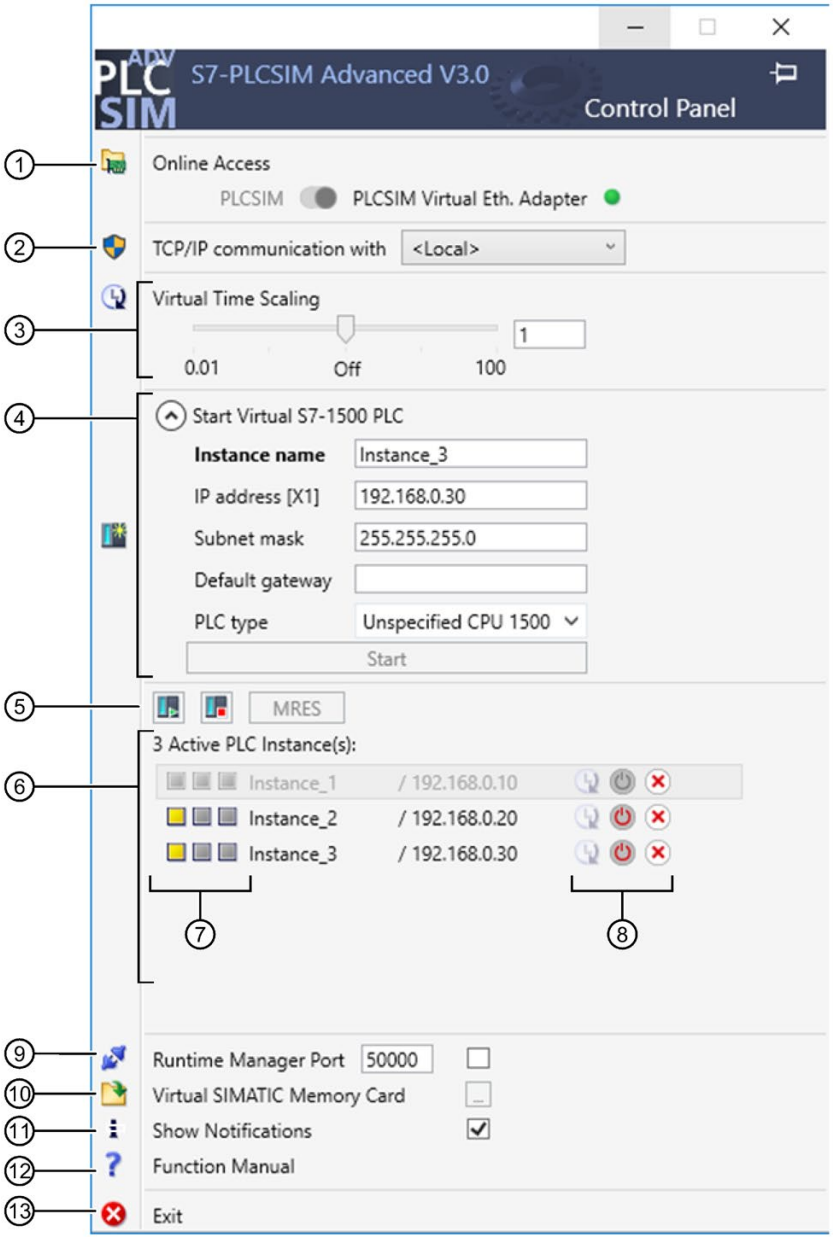
① Legt das Control Panel als Symbol in der Taskleiste ab.
② Ohne Funktion. Die Größe des Fensters ist nicht veränderbar.
③ Schließt das Control Panel und legt es im Infobereich der Taskleiste ab.
Die Instanzen und der Simulation Runtime Prozess bleiben aktiv.
Diese Funktion unterscheidet sich damit von der Exit-Funktion .
Die Exit-Funktion schaltet die lokalen Instanzen aus, meldet sie ab und schließt das Control Panel.
④ Fixiert das Control Panel auf dem Bildschirm, so dass es im Vordergrund bleibt.

Bild 5-4 Control Panel: Titelleiste

5.1.2.3 S7-PLCSIM Advanced Control Panel

Das Control Panel steht in der Version V3.0 auf Englisch zur Verfügung.

Aufbau



①	Online Zugang	Schalter zur Auswahl der Kommunikationsschnittstelle
②	TCP/IP-Kommunikation	Auswahl der Netzwerkkarte für die verteilte Kommunikation
③	Virtuelle Zeit	Schieberegler zum Einstellen des Skalierfaktors

④	**Start Virtual S7-1500 PLC**	Öffnet und schließt die Eingabefelder zum Erstellen der Instanz (Virtueller Controller).
	• Name der Instanz	Hier geben Sie einen eindeutigen Namen für die Instanz ein. Geben Sie mindestens 3, maximal 64 Zeichen ein. Wenn der Name im Netz eindeutig ist, wird die Schaltfläche "Start" aktiviert.
	• IP-Adresse • Subnetzmaske • Standard-Gateway	Die Eingabefelder werden sichtbar, wenn Sie die Kommunikationsschnittstelle auf "PLCSIM Virtual Ethernet Adapter" umschalten. Die IP-Adresse wird automatisch eingetragen.
	• CPU-Typ	Hier selektieren Sie den zu simulierenden CPU-Typ.
	• Schaltfläche "Start"	Mit der Schaltfläche erzeugen und starten Sie die Instanz.
⑤	Schaltflächen	Schaltflächen zum Bedienen der selektierten Instanzen.
⑥	Instanz-Liste	Die Liste zeigt die **lokal** vorhandenen Instanzen. Die Instanzen lassen sich mit dem Mauszeiger umsortieren.
⑦	LED-Anzeigen	Die Bedeutung der LED wird angezeigt, wenn Sie den Mauszeiger darüber bewegen.
⑧	Symbole	Symbole zur Bedienung der Instanz
⑨	Runtime Manager Port	Hier öffnen Sie einen Port auf dem lokalen PC.
⑩	Virtual SIMATIC Memory Card	Hier öffnen Sie ein Explorer-Fenster, in dem Sie den Pfad zur virtuellen Speicherkarte auswählen.
⑪	Meldungen anzeigen	Hier deaktivieren Sie die PLCSIM Advanced Meldungen in der Windows Taskleiste für die Dauer der Bedienung.
⑫	Funktionshandbuch	Hier öffnen Sie das Funktionshandbuch S7-PLCSIM Advanced im Standard PDF-Viewer.
⑬	Exit	Exit meldet alle Instanzen ab und schließt das Control Panel.

Bild 5-5 Control Panel V3.0

Schalter für Kommunikationsschnittstelle

Mit dem Schalter wählen Sie für alle zu erzeugenden Instanzen die Kommunikationsschnittstelle aus:

- "PLCSIM" entspricht der lokalen Kommunikation über Softbus (Voreinstellung).
- "PLCSIM Virtual Ethernet Adapter" entspricht der Kommunikation über TCP/IP.

Die Einstellung gilt für alle weiteren Instanzen. Die gewählte Kommunikationsschnittstelle, mit der eine Instanz startet, wird solange genutzt, bis alle Instanzen heruntergefahren werden.

Wenn bereits eine Instanz gestartet ist, dann gibt diese "ihre" Kommunikationsschnittstelle als Voreinstellung für weitere Instanzen vor.

Um die Kommunikationsschnittstelle zu wechseln, schalten Sie alle Instanzen aus und aktivieren Sie die andere Schnittstelle.

TCP/IP-Kommunikation

Sie können im laufenden Betrieb aus der Klappliste eine reale Netzwerkkarte auswählen. Sie aktivieren damit den PLCSIM Virtual Switch und stellen eine TCP/IP-Kommunikation zwischen den Instanzen und dem realen Netzwerk her.

Die Einstellung <Local> deaktiviert den PLCSIM Virtual Switch und trennt die Instanzen vom realen Netzwerk. Es ist dann nur die lokale TCP/IP-Kommunikation über den virtuellen Adapter möglich.

Virtuelle Zeit

Mit dem Schieberegler oder dem Mausrad wählen Sie den Skalierfaktor für die virtuelle Zeit.

Der eingestellte Skalierfaktor gilt für die Instanzen, für die die virtuelle Zeit aktiviert ist.

Ein Mausklick auf "Off" stellt die Voreinstellung (1) wieder her. Weitere Informationen siehe Virtuelle und reale Zeit (Seite 88).

Instanz erstellen (lokal) und starten

Um eine Instanz zu erstellen, geben Sie unter "Instance Name" einen eindeutigen Namen ein. Wenn der Name bereits im Verzeichnis der Virtual SIMATIC Memory Card existiert, dann wird diese bereits vorhandene Instanz gestartet.

In der "PLC-Type"-Klappliste wählen Sie den Typ aus: "Unspecified CPU 1500" oder "Unspecified ET 200SP CPU". Mit der Schaltfläche "Start" erzeugen und starten Sie diese Instanz.

Mit dem ersten Download aus dem TIA Portal wird die Instanz / der Virtuelle Controller getauft.

Instanz-Liste

Die Liste enthält die Instanzen, die auf dem PC oder der Virtualisierungsplattform lokal vorhanden sind. Instanzen, die bereits über die Runtime API gestartet wurden, werden erkannt und in der Liste angezeigt.

Mit den Schaltflächen "RUN" und "STOP" wählen Sie die Betriebsart der Instanz. Markieren Sie dazu eine oder mehrere Instanzen. Mit der Schaltfläche "MRES" führen Sie Urlöschen durch.

Die LED-Anzeigen zeigen den Status der Instanz, sie entsprechen denen der Hardware-CPU.
RUN und STOP werden abhängig vom aktuellen Betriebszustand der Instanz angezeigt.

Über Symbole können Sie die Instanz "bedienen":

Skalierfaktor für die virtuelle Zeit übernehmen, virtuelle Zeit deaktivieren,

Instanz einschalten ("`PowerOn`"), Instanz abschalten ("`PowerOff`"),

Instanz abschalten und vom Runtime Manager abmelden ("`Unregister`")

Runtime Manager Port

Über den eingestellten Port kann eine Remote-Verbindung zu einem weiteren Runtime Manager hergestellt werden. Der Wert muss größer als 1024 sein.

Wenn Sie das Optionskästchen aktivieren, bleibt der Port gespeichert. Sie können die Remote-Verbindung nutzen, ohne bei jedem Start des Control Panels diese Einstellung vornehmen zu müssen. Um diese Funktionalität zu nutzen, muss das Control Panel gestartet sein und im Hintergrund laufen.

Virtual SIMATIC Memory Card

In der Virtual SIMATIC Memory Card werden das Anwenderprogramm, die Hardware-Konfiguration und die remanenten Daten gespeichert. Über die Schaltfläche öffnen Sie ein Explorer-Fenster, in dem Sie den Pfad zur virtuellen Speicherkarte auswählen oder in dem der Pfad angezeigt wird.

Meldungen anzeigen

Bei jedem Start des Panels werden Hilfen und Meldungen zum Control Panel angezeigt, z. B. bei Änderung der IP-Adresse oder bei fehlender Lizenz. Deaktivieren Sie die Anzeige, wenn Sie die Meldungen nicht benötigen.

Exit - Alle Instanzen abmelden

- Der Befehl schaltet alle lokalen Instanzen auf dem PC oder der VM aus, meldet sie vom Runtime Manager ab und schließt das Control Panel.
- Der Befehl beendet den Runtime Manager, wenn keine Remote-Verbindungen zu anderen Runtime Managern bestehen.
- Wenn der Runtime Manager Remote-Verbindungen zu Instanzen auf weiteren PCs hat, dann laufen diese Instanzen und der Runtime Manager weiter.

5.1.2.4 Instanzen importieren

Voraussetzung

Die Funktion steht Ihnen nur zur Verfügung, wenn Sie das Control Panel nicht mit Admin-Rechten starten.

Instanzen importieren

Mit der Drag and Drop-Funktion können Sie Instanzen aus einem Ordner direkt in die Instanz-Liste des Control Panels importieren.

1. Öffnen Sie einen Ordner mit Instanzen, z. B. über die Schaltfläche "Virtual SIMATIC Memory Card".
2. Wählen Sie eine einzelne oder mehrere Instanzen und ziehen Sie die Instanzen auf den markierten Bereich.

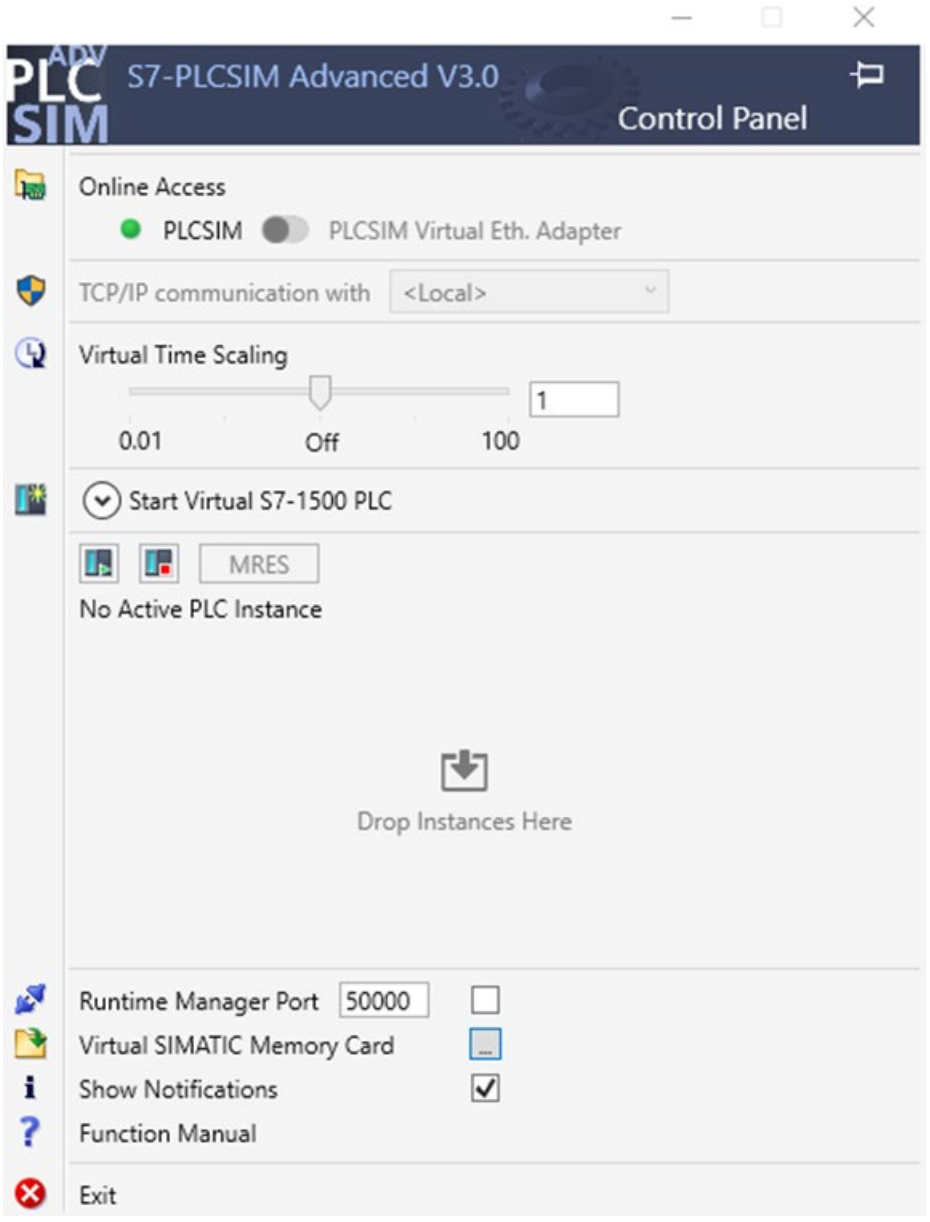


Bild 5-6 Control Panel: Instanzen importieren

5.1.3 Download

Voraussetzung

Sie können das STEP 7-Projekt auf den virtuellen Controller laden, wenn folgende Bedingungen erfüllt sind:

- Die Instanz ist über das Control Panel erzeugt.
- Das Optionskästchen "Beim Übersetzen von Bausteinen Simulierbarkeit unterstützen" ist aktiviert.

Kommunikationsschnittstelle auswählen

Im Download Dialog wählen Sie die PG/PC-Schnittstelle aus:

- "PLCSIM" für den Download über Softbus
- "Siemens PLCSIM Virtual Ethernet Adapter" für den Download über TCP/IP
- Für die verteilte Kommunikation den realen Adapter, der mit dem Netzwerk verbunden ist

Anzeigen im Download Dialog

Der Dialog in STEP 7 zeigt beim ersten Download einer CPU die kompatiblen PLCSIM Advanced Instanzen.

Wenn die Instanz noch nicht konfiguriert wurde, ist nach dem ersten Download nur **eine** Schnittstelle sichtbar und sie erscheint mit dem Gerätetyp "CPU-1500 Simulation".

Wenn die Instanz konfiguriert wurde, dann werden so viele Schnittstellen sichtbar, wie der CPU-Typ hat.

Die Lifelist zeigt die Schnittstellen einer Instanz mit ihren IP-Adressen.

Download durchführen

1. Wählen Sie die PG/PC-Schnittstelle aus.
2. Klicken Sie auf "Laden".

   → Im Fenster "Vorschau laden" zeigt STEP 7 die Meldung "Die Downloads werden auf eine simulierte CPU durchgeführt".

   → Nach dem ersten Download wird die PLCSIM Advanced Instanz mit dem CPU-Typ angezeigt.

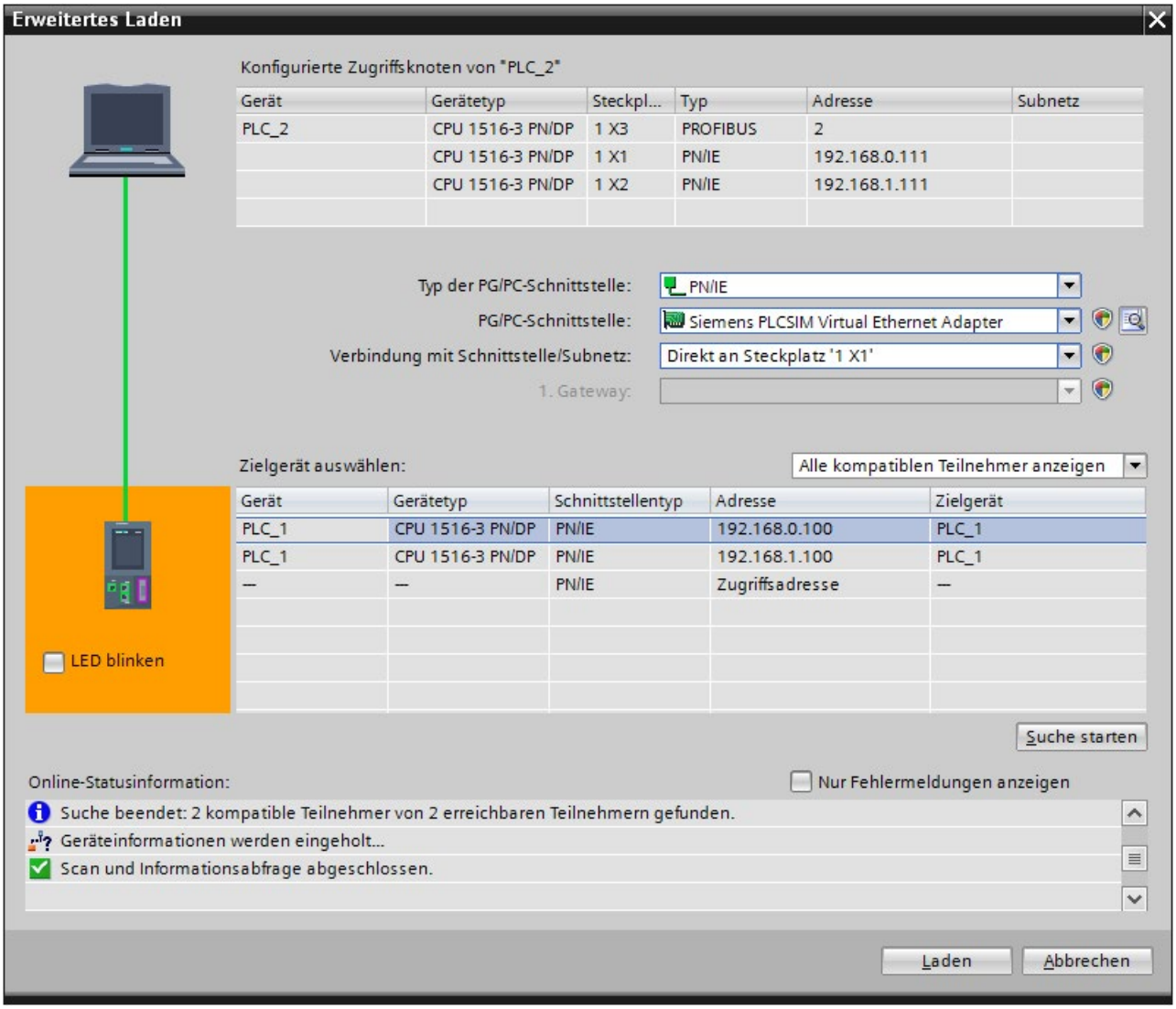


Bild 5-7 Beispiel: Download über den "PLCSIM Virtual Ethernet Adapter" (TCP/IP) nach der Taufe

5.1.4 Netzwerk-Adressen in der Simulation

5.1.4.1 Siemens PLCSIM Virtual Ethernet Adapter

IP-Adresse

Beim PLCSIM Virtual Ethernet Adapter können Sie eine statische IP-Adresse zuweisen oder über DHCP eine IP-Adresse beziehen (Voreinstellung).

MAC-Adresse

Bei der Installation des PLCSIM Virtual Ethernet Adapters wird diesem eine zufällig generierte MAC-Adresse zugewiesen.

PLCSIM Advanced verwendet nur MAC-Adressen, die als "lokal verwaltet" gekennzeichnet sind (Bit 2 in LSB).

Das Siemens-spezifische Präfix lautet: 02-1B-1B

Es folgen 3 Bytes, die nach dem Zufallsprinzip ermittelt werden.

Speicherort

Diese MAC-Adresse ist im Registry-Key "PlcsimvminiMacAddress" gespeichert.

Sie können diesen Wert überschreiben.

5.1.4.2 PLCSIM Advanced Instanzen

CPUs und Instanzen erkennen

Wenn Ethernet-Schnittstellen von CPUs und PLCSIM Advanced Instanzen in einem Netzwerk gemischt sind, dann sind die Instanzen am Suffix "PLCSIM" des Stationstyps erkennbar.

Aufbau der MAC-Adresse einer Instanz

Das folgende Bild zeigt den Aufbau der dynamisch generierten, lokal verwalteten MAC-Adresse:

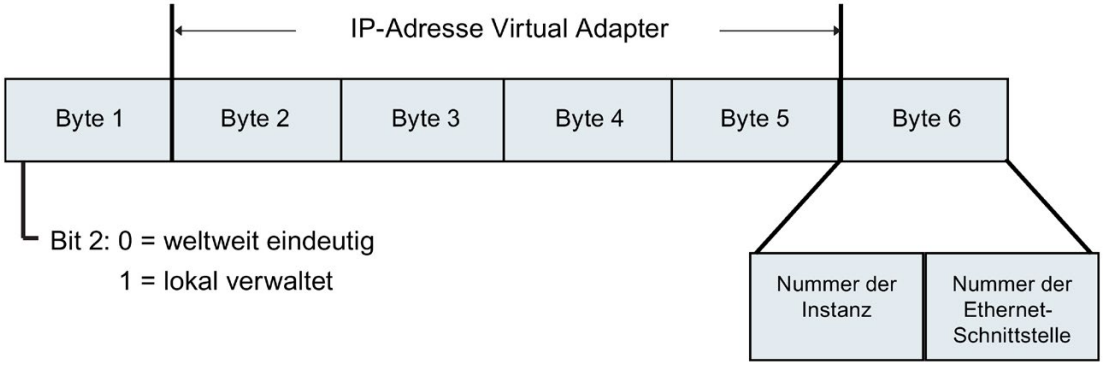


Bild 5-8 Aufbau der MAC-Adresse einer Instanz

Über die MAC-Adresse ist erkennbar, auf welchem PC eine PLCSIM Advanced Instanz gestartet ist.

Zuordnung der Ethernet-Schnittstellen

In PLCSIM Advanced V3.0 können Port-Konfigurationen der Ethernet-Schnittstellen nicht simuliert werden. Topologische Verschaltung wird nicht unterstützt. Intern wird pro Ethernet-Schnittstelle eine MAC-Adresse für einen Port reserviert.

Tabelle 5- 1 Zuordnung der Ethernet-Schnittstellen, Beispiel für eine CPU 1518-4 PN/DP

Ethernet-Schnittstelle	Letzte Stelle der MAC-Adresse
IE 1 IE 1 / Port 1	.....0 1
IE 2 IE 2 / Port 1	.....2 3
IE 3 IE 3 / Port 1	.....4 5

Beispiel

02-C0-A8-00-83-10 bedeutet:

02 → lokal verwaltete MAC-Adresse einer PLCSIM Advanced Instanz

C0-A8-00-83 → IP des Siemens PLCSIM Virtual Ethernet Adapters = 192.168.0.131

1 → Instanz 1

0 → Ethernet-Schnittstelle IE 1

Wenn beim Hochlauf der PLCSIM Advanced keine Virtual SIMATIC Memory Card geladen wurde, dann werden die Schnittstellen der PLCSIM Advanced Instanzen mit ihrer lokal verwalteten MAC-Adresse angezeigt.

5.1.5 Peripherie-I/O simulieren

Die Runtime API schreibt in einen und liest aus einem Speicherbereich. Dieser Speicher wird am Zykluskontrollpunkt und beim Aufruf zyklischer und azyklischer OBs (Teilprozessabbilder, Alarme, Ereignisse) mit dem internen Prozessabbild des virtuellen S7-1500 Controllers synchronisiert. Die direkten Peripheriezugriffe erfolgen auf diesen Speicherbereich. Es kann jeweils nur ein Prozess auf diesen Speicher zugreifen.

Der virtuelle Controller muss sich in RUN befinden, um Änderungen zu übernehmen, die die API vorgibt.

Hinweis

Dominanz der API beim Synchronisieren

Beim Synchronisieren dominiert die API. Wenn das Anwenderprogramm auf denselben Adressbereich schreibt wie die API, dann überschreiben die Änderungen der API jene des virtuellen Controllers.

Siehe auch

Abweichende E/A-Werte im STEP 7-Anwenderprogramm (Seite 404)

5.1.6 Kommunikation simulieren

5.1.6.1 Simulierbare Kommunikationsdienste

PLCSIM Advanced V3.0 unterstützt folgende Kommunikationsmöglichkeiten:

Tabelle 5- 2 Unterstützte Kommunikationsmöglichkeiten

Möglichkeiten der Kommunikation	Funktionalität / Anweisungen
PG-Kommunikation	Zur Inbetriebnahme, Test, Diagnose
Offene Kommunikation über TCP/IP	• TSEND_C / TRCV_C • TSEND / TRCV • TCON[1, 3] • T_DISCON
Offene Kommunikation über ISO-on-TCP	• TSEND_C / TRCV_C • TSEND / TRCV • TCON • T_DISCON
Offene Kommunikation über UDP[2]	• TUSEND / TURCV • TCON • T_DISCON
Kommunikation über Modbus TCP[2]	• MB_CLIENT • MB_SERVER
E-Mail[2, 3]	• TMAIL_C
S7-Kommunikation	• PUT / GET • BSEND / BRCV • USEND / URCV
OPC UA Server[2]	Datenaustausch mit OPC UA Clients
Webserver[2, 3]	Datenaustausch über HTTP

1 Wenn die Schnittstelle "PLCSIM" (Softbus) eingestellt ist, wird die Kommunikation **intern** über Iso-On-TCP geführt.

2 Nur über die Kommunikationsschnittstelle "PLCSIM Virtual Ethernet Adapter" (TCP/IP). "Zugriff auf PLC über Kommunikationsmodul" wird nicht unterstützt.

3 Gesicherte TCP-Kommunikation wird nicht unterstützt.

Für die Kommunikation mit TUSEND / TURCV gelten Besonderheiten, siehe Einschränkungen bei Kommunikationsdiensten (Seite 399).

TMAIL_C

Bei der Verwendung der Anweisung TMAIL_C darf sich der Mailserver nicht auf dem gleichen PC wie die PLCSIM Advanced Instanz befinden.

Abhilfe

Stellen Sie den Mailserver über einen anderen PC im Netzwerk zur Verfügung.

5.1.6.2 Kommunikation zwischen Instanzen

PLCSIM Advanced unterstützt die Kommunikation zwischen Instanzen. Eine Instanz kann eine Simulation in PLCSIM Advanced V2.0 oder eine Simulation in WinCC Runtime ab V14 sein.

Sie können zwei Instanzen von PLCSIM Advanced ausführen, die dann untereinander kommunizieren. Damit Instanzen untereinander kommunizieren können, benötigen sie eine eindeutige IP-Adresse.

Jede simulierte CPU benötigt eine eindeutige IP-Adresse

Wenn die CPUs die gleiche IP-Adresse haben, können Sie nicht mehrere Simulationen ausführen. Jede simulierte CPU benötigt eine eindeutige IP-Adresse.

Vergewissern Sie sich, dass die IP-Adressen in STEP 7 einmalig sind, bevor Sie Ihre Simulationen starten.

T-Bausteinanweisungen und UDP

PLCSIM Advanced simuliert T-Bausteinverbindungen, für die das UDP-Protokoll konfiguriert ist, nur über die Kommunikationsschnittstelle "PLCSIM Virtual Ethernet Adapter" (TCP/IP).

T-Bausteinanweisungen und Datensegmentierung

PLCSIM Advanced implementiert T-Bausteinanweisungen mit einer Datensegmentierung von 4 kByte. Eine reale CPU hat eine Datensegmentierung von 8192 Byte.

Wenn Sie mehr als 4 kByte in einer einzelnen Anweisung TSEND senden und im Ad-hoc-Modus mit einer Anweisung TRCV Daten empfangen, erzeugt die Anweisung TRCV neue Daten mit nur 4 kByte. Sie müssen die Anweisung TRCV mehrmals ausführen, um weitere Bytes zu empfangen.

5.1.7 Projektdaten offline für die Simulation bereitstellen

Simulationen unabhängig von STEP 7

Um unabhängig von STEP 7 Simulationen durchzuführen, können Sie das Anwenderprogramm und die HW-Konfiguration in STEP 7 in einem Verzeichnis speichern.

Remanente Daten sicher speichern

Die remanenten Daten werden automatisch beim Herunterfahren der Virtuellen Controller gespeichert.

Um die remanenten Daten sicher in der Virtual SIMATIC Memory Card zu speichern, müssen die Instanzen korrekt abgemeldet werden. Nutzen Sie dazu eine der folgenden Funktionen:

- Die API-Funktion `PowerOff()`
- Im Control Panel die Funktion "Instanz abschalten"  , "Instanz abmelden"  oder die Exit-Funktion "Alle Instanzen abmelden"

Projektdaten offline bereitstellen

1. Legen Sie in STEP 7 in der Projektnavigation für die CPU im Ordner "Card Reader/USB-Speicher" einen "Benutzerdefinierten Card Reader" für Ihre Projektdaten an.
2. Wählen Sie im Dialog "Vorschau Laden" für das Zielgerät als Aktion "PLC Simulation Advanced", klicken Sie dazu in das Auswahlfeld.
   → Das Projekt wird im Verzeichnis <Virtual Memory Card>\SIMATIC.S7S\OMSSTORE gespeichert.
3. Speichern Sie den Ordner "\SIMATIC.S7S" mit den Projektdaten auf ein Medium Ihrer Wahl.

Bild 5-9 Card Reader hinzufügen

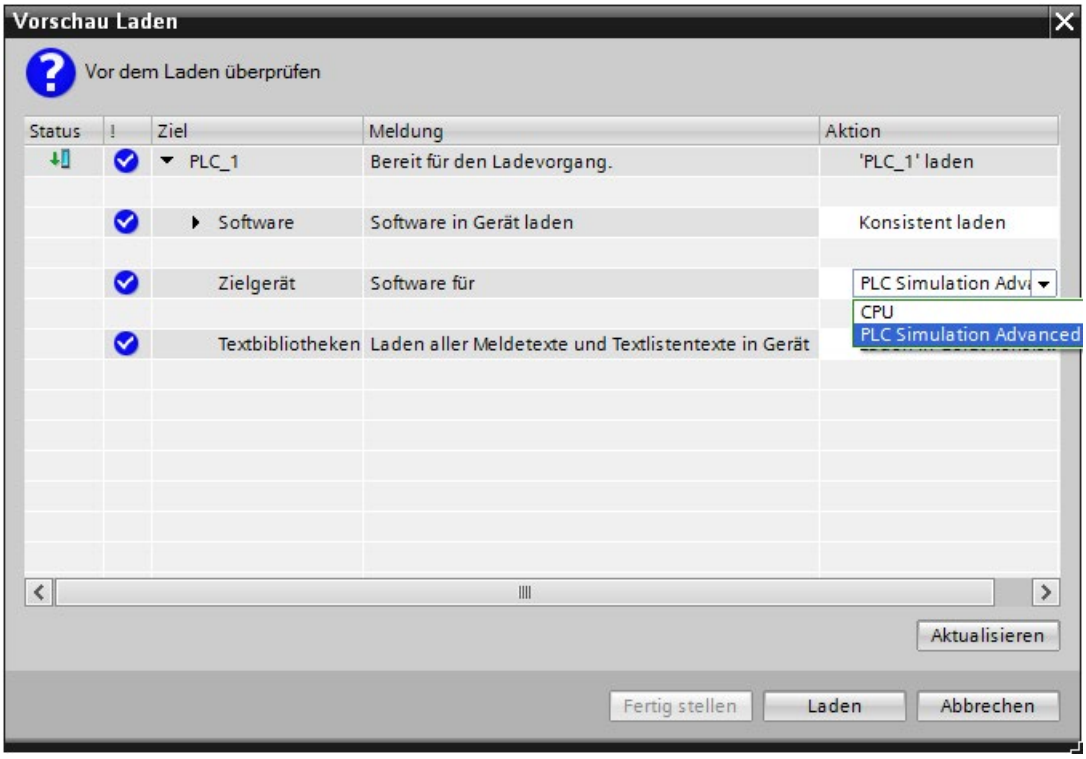


Bild 5-10 Vorschau Laden Dialog

Projektdaten für die Simulation bereitstellen

1. Erstellen Sie auf dem PC, auf dem PLCSIM Advanced installiert ist, in dem Verzeichnis, in dem die Instanz ihre Daten speichert, das Verzeichnis "\SIMATIC_MC".
2. Verschieben Sie den Ordner "\SIMATIC.S7S" in das erstellte Verzeichnis.
   → Die Instanzen können mit den Projektdaten gestartet werden.

API-Funktionen

Über die Anwenderschnittstellen können Sie die Projektdaten für eine Instanz nutzen. Verwenden Sie dazu folgende Funktionen:

API-Funktionen

- GetStoragePath() (Seite 163)
- StoragePath { get; set; } (Seite 164)
- ArchiveStorage() (Seite 165)
- RetrieveStorage() (Seite 166)

Siehe auch

Controller - Informationen und Einstellungen (Seite 157)

5.2 CPU mit ODK-Funktionalität simulieren

Einleitung

Das ODK ist ein Engineering-Werkzeug, das die Erstellung von Hochsprachenanwendungen für S7-1500 CPUs ermöglicht. Sie erstellen damit Funktionsbibliotheken, die im STEP 7-Anwenderprogramm genutzt werden.

Das ODK für PLCSIM Advanced V3.0 unterstützt die Programmiersprache C++.

Die Beschreibung zu ODK finden Sie im "Programmier- und Bedienhandbuch "S7-1500 Open Development Kit 1500S" ab V2.5 Ausgabe 12/2017: SIMATIC STEP 7 (TIA Portal) Optionen ODK 1500S (https://support.industry.siemens.com/cs/document/109752687)

Relevant für ODK-Anwendungen unter PLCSIM Advanced ist dabei das Kapitel 5 "Entwicklung einer CPU Funktionsbibliothek für die Echtzeit-Umgebung".

Unterstützte CPUs

PLCSIM Advanced V3.0 unterstützt die ODK-Funktionalität der folgenden Controller:

- CPU 1518(F)-4 PN/DP ODK
- CPU 1518(F)-4 PN/DP MFP

5.2.1 Besonderheiten bei ODK

CPU mit ODK-Funktionalität mit PLCSIM Advanced simulieren

Die Simulation einer CPU mit ODK-Funktionalität erfordert ein besonderes Vorgehen beim Start.

Sie haben dazu folgende Möglichkeiten:

- Starten Sie die Instanzen von einer Virtual SIMATIC Memory Card, die die Projektdaten für die CPU mit ODK-Funktionalität enthält.
- Wählen Sie vor dem Starten der Instanzen den CPU-Typ über die API aus, z. B. "CPU1518MFP".
- Wählen Sie nach dem ersten Download im Control Panel die Funktionen "Instanz abschalten" und "Instanz einschalten" .

Hinweis

Wenn Sie über das PLCSIM Advanced Control Panel den ersten Download auf eine CPU mit dem Typ "Unspecified CPU 1500" durchführen, dann wird auf der Virtual SIMATIC Memory Card kein ODK1500S-Verzeichnis erzeugt. Die CPU lässt sich nicht nach RUN schalten. Im Diagnosepuffer finden Sie in diesem Fall Meldungen zu fehlenden ODK-Bausteinen (z. B. SFC 2013).

Unterstützte Funktionsbibliotheken

PLCSIM Advanced V3.0 unterstützt die folgenden Funktionsbibliotheken für die Echtzeit-Umgebung:

- CPU Funktionsbibliothek: Original Shared Object, SO-Datei wie für die Hardware CPUs
- PLCSIM Advanced Funktionsbibliothek (Windows Sync):
  - eine 32-Bit Windows DLL für ODK Runtime
  - eine 64-Bit Windows DLL für ODK Runtime

Hinweis

Funktionsbibliotheken nicht mischen

Bei der Simulation mit PLCSIM Advanced dürfen jeweils nur Funktionsbibliotheken mit gleichem Binärformat geladen sein.

Wenn Sie Funktionsbibliotheken mit einem anderen Binärformat nutzen wollen, müssen Sie zuvor alle anderen entladen.

Hinweis

Kein Know-How-Schutz bei SO-Dateien

Die SO-Dateien bei ODK sind nicht know-how-geschützt.

Simulation der ODK-Anwendung bei PLCSIM Advanced

Wenn Sie das TIA Projekt auf die PLCSIM Advanced geladen haben und die Anweisung "*<STEP7Prefix>*_Load" zum ersten Mal aufgerufen wurde, startet jede PLCSIM Advanced Instanz einen weiteren Windows-Prozess ("ODK-Client"), in dem die ODK-Anwendung synchron zum STEP 7-Anwenderprogramm ausgeführt wird.

Welcher ODK-Client gestartet wird, hängt von der Funktionsbibliothek ab, die geladen werden soll:

- "Siemens.Simatic.PlcSim.Vplc1500.ODKClient.**so**.exe" für ein original Shared Object
- "Siemens.Simatic.PlcSim.Vplc1500.ODKClient.**x86**.exe" für eine 32 Bit-Anwendung
- "Siemens.Simatic.PlcSim.Vplc1500.ODKClient.**x64**.exe" für eine 64 Bit-Anwendung

Die ausführbaren Dateien dieser Prozesse befinden sich im gleichen Verzeichnis wie die der PLCSIM Advanced Instanzen ("Siemens.Simatic.Simulation.Runtime.Instance.exe").

Hinweis

PLCSIM Advanced unterstützt keine asynchronen ODK-Funktionen.

Fehlercodes

Für die Anweisungen in der Echtzeit-Umgebung gelten die Fehlercodes, die auch im Programmier- und Bedienhandbuch "S7-1500 Open Development Kit 1500S" beschrieben sind. Zusätzlich gibt es Fehlercodes für PLCSIM Advanced, da die ODK-Client-Prozesse unerwartet geschlossen werden können und dadurch ein Fehlerhandling erforderlich ist.

Einschränkungen für die Stack-Bearbeitung

Hinweis

Einschränkungen für die Stack-Bearbeitung bei der Ausführung von CPU Funktionsbibliotheken für die Echtzeit-Umgebung

PLCSIM Advanced ignoriert die Stack-Größe für eine CPU Funktionsbibliothek, die über den Parameter <SyncCallStackSize> eingestellt wird. PLCSIM Advanced stellt immer die maximale Stack-Größe von 1 MB zur Verfügung.

Siehe Programmier- und Bedienhandbuch "S7-1500 Open Development Kit 1500S" V2.5, Kapitel 5.1.4 Ablaufeigenschaften einer CPU Funktionsbibliothek definieren.

PLCSIM Advanced kann keine Exceptions vom Typ "Stack Overflow" fangen, wenn CPU Funktionsbibliotheken für die Echtzeit-Umgebung (SO-Dateien) ausgeführt werden.

Sorgen Sie bei der Entwicklung einer CPU Funktionsbibliothek (SO-Datei) dafür, dass die maximale Stack-Größe von 1 MB nicht überschritten wird. Ein Überlauf des Stacks führt zu einem nicht definierten Verhalten und kann zum Beenden des ODK-Client-Prozesses führen.

Hinweis

Einschränkungen für die Heap-Bearbeitung bei der Ausführung von CPU Funktionsbibliotheken (Windows Sync)

Wenn bei der Ausführung einer C/C++ Funktion aus einer CPU Funktionsbibliothek (DLL-Datei) eine Heap Korruption auftritt, dann wird dieser Programmfehler zunächst ignoriert und die Ausführung der Funktion fortgesetzt. Erst nach vollständiger Abarbeitung der Funktion wird der entsprechende Fehlercode zurückgegeben (0x8090).

Sorgen Sie bei der Entwicklung einer CPU Funktionsbibliothek (DLL-Datei) dafür, dass Heap Korruptionen vermieden werden. Damit stellen Sie sicher, dass nach vollständiger Abarbeitung einer C/C++ Funktion kein Fehlercode zurückgegeben wird.

5.2.2 Funktionen laden

Funktionen laden - Anweisung "*<STEP7Prefix>*_Load"

Wenn Sie das TIA Projekt auf die PLCSIM Advanced geladen haben und die Anweisung "*<STEP7Prefix>*_Load" zum ersten Mal aufgerufen wurde, startet jede PLCSIM Advanced Instanz einen weiteren Windows-Prozess. Der ODK-Client versucht dann, die Funktionsbibliothek zu laden, die in der SCL-Datei spezifiziert ist. Diese liegt im Verzeichnis "<storage path of the instance> \SIMATIC_MC\ODK1500S". Siehe `GetStoragePath()`, `SetStoragePath()` im Kapitel Controller - Information und Einstellungen (Seite 163).

Der ODK-Client-Prozess dauert solange an, bis die Anweisung "*<STEP7Prefix>*_Unload" aufgerufen wird, um die zuletzt geladene Funktionsbibliothek zu entladen, oder bis der Prozess der PLCSIM Advanced Instanz endet.

Der Funktionsaufruf ist synchron und kehrt nach Abschluss der Operation zurück. Der Ausgangsparameter informiert dabei über den Status des Fortschritts.

ODK-Fehlercodes bei PLCSIM Advanced

Die folgende Tabelle enthält die Fehlercodes, die neben den für die CPU geltenden Fehlercodes speziell für ODK-Anwendungen bei PLCSIM Advanced gelten:

Tabelle 5- 3 ODK: Ausgangsparameter - Funktionen laden

DONE	BUSY	ERROR	STATUS	Beschreibung
0	0	1	0x80A4 = -32604	• Der ODK-Client-Prozess kann nicht gestartet werden. • Eine Verbindung zum ODK-Client kann nicht hergestellt werden oder wurde unterbrochen.
0	0	1	0x8095 = -32619	• Der aktuell laufende ODK-Client-Prozess erwartet eine Funktionsbibliothek mit einem anderen Binärformat.

5.2.3 Funktionen aufrufen

Funktionen aufrufen - Anweisung "*<STEP7Prefix>*SampleFunction"

Beim Aufruf von ODK-Funktionen werden Daten zwischen dem virtuellen Controller und der Funktionsbibliothek ausgetauscht.

Die Ausführung einer einzelnen Funktion kann durch die Ausführung von höher priorisierten OBs unterbrochen werden.

Die Ausführung einer Funktion ist technisch eine asynchrone Operation, da sie in einem anderen Prozess ausgeführt wird. Die Prozesse werden aber über den virtuellen Controller synchronisiert. Das bedeutet, dass der Funktionsaufruf nicht zurückkehrt, bevor entweder die Funktion zurückkehrt oder der ODK-Client-Prozess während der Ausführung geschlossen wird.

ODK-Fehlercodes bei PLCSIM Advanced

Die folgende Tabelle enthält die Fehlercodes, die neben den für die CPU geltenden Fehlercodes speziell für ODK-Anwendungen bei PLCSIM Advanced gelten:

Tabelle 5- 4 ODK: Ausgangsparameter - Funktionen aufrufen

DONE	BUSY	ERROR	STATUS	Beschreibung
0	0	1	0x80A4 = -32604	• Die Verbindung zum ODK-Client wurde unterbrochen.

5.2.4 Funktionen entladen

Funktionen entladen - Anweisung "*<STEP7Prefix>*_Unload"

Die CPU Funktionsbibliothek wird mit dem Aufruf der Anweisung "*<STEP7Prefix>*_Unload" entladen. Wenn keine weitere Funktionsbibliothek geladen wird oder wenn der Prozess der PLCSIM Advanced Instanz geschlossen wird, dann wird der ODK-Client-Prozess heruntergefahren.

Der Funktionsaufruf ist asynchron, der Aufruf kehrt sofort zurück. Der Ausgangsparameter informiert dabei über den Status des Fortschritts.

5.3 Motion Control simulieren

Einschränkungen

PLCSIM Advanced simuliert die reale CPU, nicht aber projektierte, verbundene Technologiemodule oder andere Peripherie.

Der Download eines STEP 7-Projekts mit Technologiemodulen für den Betrieb von Motion Control ist möglich. Die integrierte Logik der Technologiemodule ist aber nicht Teil der Simulation. Daher werden auch die dazugehörigen Motion Control-Anweisungen nicht unterstützt.

OB 91 und OB 92

Wenn Sie ein Motion Control Projekt, das den OB 91 und OB 92 enthält, von STEP 7 V13 konvertieren, dann können Sie dieses Projekt nicht auf eine PLCSIM Advanced laden.

Abhilfe

Löschen Sie im Projekt den OB 91 und den OB 92 und übersetzen Sie das Projekt erneut.

Die OBs werden dadurch mit dem für PLCSIM Advanced erforderlichen Simulations-Support neu erzeugt.
Das Übersetzen setzt die Eigenschaften der Bausteine auf Standardwerte zurück.
Stellen Sie die erforderlichen Einstellungen in den Eigenschaften wieder her.

Überlauf von Motion Control OBs

Aufgrund der geringen Leistung von PCs kann es vorkommen, dass ein neuer Motion Control OB gestartet wird, bevor der vorherige fertig berechnet wurde. Die CPU kann dadurch in den Betriebszustand STOP versetzt werden. In der Diagnose finden Sie nur den Hinweis, dass ein Wechsel in den Betriebszustand STOP erfolgt ist.

Abhilfe

Verlangsamen Sie den Ablauf der virtuellen Zeit, um dem OB mehr Zeit zur Bearbeitung zu geben.

Informationen zum Skalierfaktor finden Sie in Kapitel Simulation beschleunigen und verlangsamen (Seite 90).

Simulation mit externer Simulations-Software

Hinweis

Bei einem virtuellen S7-1500 Controller sind die Technologieobjekte mit dem Prozessabbild verschaltet. Simulations-Software kann dadurch über die Anwenderschnittstellen (API) von PLCSIM Advanced auf das Prozessabbild zugreifen und darüber das Verhalten der sonst angeschlossenen Achsen simulieren.

Simulations-Modus in STEP 7

Der Simulations-Modus in STEP 7 ist eine Standardfunktion der Technologieobjekte und ist unabhängig von PLCSIM Advanced.

Wenn Sie eine Achse im Simulationsbetrieb verfahren möchten, aktivieren Sie in STEP 7 unter "Technologieobjekt > Konfiguration > Grundparameter > Simulation" das Optionskästchen "Simulation aktivieren". Bei einer virtuellen Achse ist hier keine weitere Einstellung erforderlich.

Rückmeldung der Achsposition

Der Geschwindigkeits-Sollwert des simulierten Antriebs wird mit einer Zeitverzögerung (PT1) zum Positions-Istwert aufintegriert. Das Ergebnis dieser Berechnung wird dem Technologieobjekt als Positions-Istwert der Achse zurückgemeldet.

Referenzpunktfahrt der Achse

Wenn Sie in STEP 7 für die Referenzpunktfahrt "Nullmarke über PROFIdrive-Telegramm verwenden" ausgewählt haben, reagiert PLCSIM Advanced sofort auf jeden aktiven (Modus 2, 3, 8) oder passiven (Modus 4, 5) Referenzpunktfahrtbefehl (MC_Home). Dabei wird die tatsächliche Position als Referenzpunkt festgelegt.

Weitere Informationen

Informationen zur "Einstellung in der Antriebs- und Geberanbindung", zur Istwertberechnung einer virtuellen Achse und zum Thema "Virtuelle Achse/Simulation" erhalten Sie im Funktionshandbuch S7-1500T Motion Control (https://support.industry.siemens.com/cs/ww/de/view/109481326).

Weitere Informationen erhalten Sie im Funktionshandbuch S7-1500 Motion Control (https://support.industry.siemens.com/cs/ww/de/view/109739589) und in den Gerätehandbüchern zu den unterstützten SIMATIC Controllern (https://support.industry.siemens.com/cs/ww/de/view/109744173)

Virtuelles Zeitverhalten

6

Der virtuelle Controller nutzt intern für die Simulation zwei Arten von Uhren: Eine virtuelle und eine reale Uhr. Basis für das Anwenderprogramm ist immer die virtuelle Uhr. Sie wird von Komponenten genutzt, die für den Ablauf des STEP 7-Anwenderprogramms relevant sind, wie zyklische OBs, Zykluszeitüberwachung, minimale Zykluszeit, virtuelle Systemzeit und Zeitberechnungen. Auch die Dauer zwischen zwei Zykluskontrollpunkten wird in virtueller Zeit gemessen.

Die virtuelle Uhrzeit kann zu Testzwecken beschleunigt oder verlangsamt werden.

Die reale Uhr läuft immer unverändert. Sie wird von Komponenten genutzt, die nicht von Steuerungsprozessen abhängig sind, z. B. die Kommunikation mit STEP 7.

Unterbrechung des Prozesses

Da PLCSIM Advanced in einer Windows-Umgebung läuft, kann es vorkommen, dass Windows den Prozess des virtuellen Controllers vorübergehend unterbricht. In so einem Fall bleiben sowohl die virtuelle als auch die reale Uhr im virtuellen Controller stehen. Sie laufen erst dann weiter, wenn Windows die Bearbeitung des Prozesses wieder fortsetzt.

Virtuelle Systemzeit

Beim Starten von PLCSIM Advanced startet die virtuelle Systemzeit des virtuellen Controllers mit der Systemzeit von Windows.

Die virtuelle Systemzeit basiert auf der virtuellen Uhr, d. h. wenn ein Skalierfaktor verwendet wird, dann läuft die Systemzeit entsprechend schneller oder langsamer.

Alle Ereignisse, die der virtuelle Controller an die API sendet, enthalten einen Zeitstempel basierend auf der Systemzeit.

Hinweis

Unterschied Systemzeit und Lokalzeit

- Systemzeit: UTC ± 0 ohne Sommerzeit / Winterzeit
- Lokalzeit: UTC ± Zeitzone mit Sommerzeit / Winterzeit

API-Funktionen

- GetSystemTime() (Seite 245)
- SetSystemTime() (Seite 245)
- SystemTime { get; set; } (Seite 246)

Zeitversatz

Hinweis

Beachten Sie, dass die Uhrzeitangaben von virtueller Systemzeit und realer Lokalzeit sich um den Zeitversatz unterscheiden, der sich zusätzlich zum gewählten Skalierfaktor aus dem Zeitzonen-Versatz und dem Sommer-/Winterzeit-Versatz bildet.

Skalierfaktor

Mit einem Skalierfaktor können Sie für Simulationen die virtuelle Uhr des virtuellen Controllers beschleunigen oder verlangsamen.

- Die Voreinstellung ist 1, d. h. der Verlauf der virtuellen Zeit entspricht dem Verlauf der realen Zeit.
- **Schnelllauf:** Ein Skalierfaktor größer 1 beschleunigt die virtuelle Uhr.

  Beispiel: Skalierfaktor 2,0 → Die virtuelle Zeit läuft zweimal so schnell.
- **Zeitlupe**: Ein Skalierfaktor kleiner 1 verlangsamt die virtuelle Uhr.

  Beispiel: Skalierfaktor 0,5 → Der Fortschritt der virtuellen Zeit verlangsamt sich auf 50%.

API-Funktionen

- GetScaleFactor() (Seite 246)
- SetScaleFactor() (Seite 247)
- ScaleFactor { get; set; } (Seite 248)

Siehe auch

Einstellungen für die virtuelle Zeit (Seite 245)

6.1 Simulation beschleunigen und verlangsamen

Einfluss von Schnelllauf und Zeitlupe

Simulationen können beschleunigt und verlangsamt werden. Schnelllauf und Zeitlupe beeinflussen nur zeitbasierte Komponenten, z. B. zyklische OBs. Im Vergleich zur realen Zeit werden sie durch Schnelllauf häufiger und durch Zeitlupe seltener ausgeführt.

Schnelllauf und Zeitlupe ändern nicht die Ausführungsgeschwindigkeit des CPU-Maschinen-Codes. Es wird z. B. nicht die Geschwindigkeit geändert, mit der alle Operationen eines OB1-Zyklus ausgeführt werden. Die Ausführungsgeschwindigkeit hängt vom Prozessor des PC ab, auf dem der virtuelle Controller läuft. Wenn Sie den Skalierfaktor ändern, werden mehr oder weniger Zykluskontrollpunkte in einer festen Zeitspanne der virtuellen Zeit erreicht.

Hinweis

Performance

Die Performance ist u. a. abhängig vom Umfang Ihres Projekts.

Wenn der Skalierfaktor zu hoch ist und die Zykluszeitüberwachung anzeigt, dass der PC nicht in der Lage war, den OB1 oder zyklische OBs in der vorgegebenen Zeit zu berechnen, geht der virtuelle Controller in STOP.

Empfehlung: Um dies zu vermeiden, beginnen Sie mit einem kleinen Skalierfaktor und tasten Sie sich schrittweise an einen Skalierfaktor heran, bei dem der virtuelle Controller in RUN bleibt.

Wenn es zu einem Überlauf von Ereignissen kommt, dann verlangsamen Sie die Geschwindigkeit der Simulation. Siehe Überwachung Überlauf (Seite 404) und Zykluskontrolle (Seite 248).

Schnelllauf

Um die virtuelle Zeit zu beschleunigen, wählen Sie im Control Panel oder in der API einen Skalierfaktor größer 1.

Zeitlupe

Um die virtuelle Zeit zu verlangsamen, wählen Sie im Control Panel oder in der API einen Skalierfaktor kleiner 1.

API-Funktionen

- GetScaleFactor() (Seite 246)
- SetScaleFactor() (Seite 247)
- ScaleFactor { get; set; } (Seite 248)

Siehe auch

Einstellungen für die virtuelle Zeit (Seite 245)

6.2 Simulation anhalten

Freeze-Zustand des virtuellen Controllers

Um eine Simulation anzuhalten und um Simulations-Partner zu synchronisieren, kann ein virtueller Controller über die API in einen Freeze-Zustand versetzt werden. Wenn der virtuelle Controller einen Synchronisationspunkt erreicht hat, sendet er das Ereignis `OnSyncPointReached` an die API-Clients.

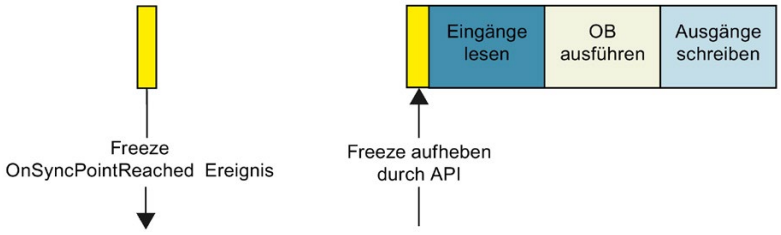


Bild 6-1 Freeze-Zustand des virtuellen Controllers

Im Freeze-Zustand tritt Folgendes ein:

- Die virtuelle Zeit wird angehalten.
- Es laufen keine OBs und keine Zeiten.
- Das Anwenderprogramm wird nicht weiter ausgeführt.
- Der virtuelle Controller ist vom TIA Portal aus noch erreichbar.
- Die Eingangs- und Ausgangsdaten des virtuellen Controllers sind in einem konsistenten Zustand.

Hinweis

Freeze-Zustand beim Download

Um einen Download im Freeze-Zustand abzuschließen, muss der virtuelle Controller am Ende des Downloads einen Zykluskontrollpunkt passieren.

Hinweis

Freeze-Zustand ≠ Betriebszustand

Der Freeze-Zustand ist ein interner Betriebszustand des virtuellen Controllers. Er entspricht nicht dem Betriebszustand RUN/STOP einer CPU. Im Freeze-Zustand behält der virtuelle Controller den letzten Betriebszustand bei.

- Die LED-Anzeige auf dem Control Panel und auf dem Webserver zeigt für die Instanz entsprechend RUN oder STOP an.
- Die Instanz zeigt den Betriebszustand `SROS_FREEZE` / `Freeze` an, siehe EOperatingState (Seite 373).

Synchronisationspunkte

Ein Synchronisationspunkt liegt immer dann vor, **bevor** Eingänge eingelesen werden, z. B. am Zykluskontrollpunkt oder am Anfang eines zyklischen OBs.

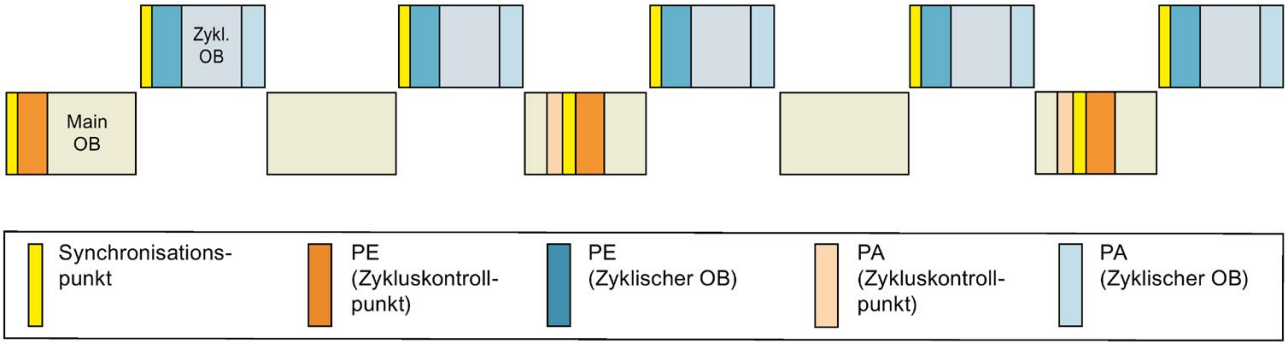


Bild 6-2 Übersicht zu den Synchronisationspunkten

Freeze-Zustand auslösen

Um den Freeze-Zustand auszulösen, stehen für den virtuellen Controller folgende Betriebsarten zur Verfügung:

- SingleStep-Betriebsarten

  Siehe Simulations-Partner zyklusgesteuert synchronisieren (Seite 93).

- TimespanSynchronized-Betriebsarten

  Siehe Simulations-Partner zeitgesteuert synchronisieren (Seite 95).

In der Betriebsart Default geht der virtuelle Controller nicht in einen Freeze-Zustand.

API-Funktionen

- Einstellungen für die Zykluskontrolle (Seite 248)
- GetOperatingMode() (Seite 248)
- SetOperatingMode() (Seite 249)
- OperatingMode { get; set; } (Seite 249)
- EOperatingMode (Seite 374)

6.3 Simulations-Partner synchronisieren

6.3.1 Simulations-Partner zyklusgesteuert synchronisieren

SingleStep-Betriebsarten

Mit den SingleStep-Betriebsarten des virtuellen Controllers werden mehrere Simulations-Partner (Clients) zyklusgesteuert synchronisiert. Die Betriebsarten definieren den Synchronisationspunkt, an dem der virtuelle Controller in den Freeze-Zustand wechselt und das Ereignis `OnSyncPointReached` sendet.

Tabelle 6- 1 Zyklusgesteuerte Betriebsarten (SingleStep)

Betriebsart	Synchronisationspunkt		Mindestzykluszeit[1]
	Zykluskontrollpunkt	Vor dem Einlesen des Teilprozessabbilds	
	"C"	"P"	"T"
`SingleStep_C`	✓		
`SingleStep_P`		✓	
`SingleStep_CP`	✓	✓	
`SingleStep_CT`	✓		✓
`SingleStep_CPT`	✓	✓	✓

[1] Zusätzlich wird in dieser Betriebsart die Mindestzykluszeit des OB 1 überschrieben. Wenn Sie über die API eine Mindestzykluszeit von 200 ms definieren, dann ist der Mindestabstand zwischen zwei Zykluskontrollpunkten 200 virtuelle Millisekunden. Die Voreinstellung ist 100 ms.

API-Funktionen / Ereignisse

- GetOverwrittenMinimalCycleTime_ns() (Seite 251)
- SetOverwrittenMinimalCycleTime_ns() (Seite 252)
- OverwrittenMinimalCycleTime_ns { get; set; } (Seite 252)
- RunToNextSyncPoint() (Seite 253)
- OnSyncPointReached (Seite 288)
- EventCallback_II_SREC_ST_UINT32_INT64_INT64_UINT32 (Seite 322) / Delegate_II_EREC_DT_UINT32_INT64_INT64_UINT32 (Seite 336)

Freeze-Zustand beenden

Die Funktion `RunToNextSyncPoint()` hebt den Freeze-Zustand auf und veranlasst den virtuellen Controller, bis zum nächsten Synchronisationspunkt weiterzulaufen.

Auch ein Wechsel in den Betriebszustand Default beendet den Freeze-Zustand.

Beispiel

Die Abbildung zeigt schematisch den Ablauf in der Betriebsart `SingleStep_CP`.

Neben dem Ereignis `OnSyncPointReached`, sendet der virtuelle Controller auch die virtuelle Zeit, seit der letzte Synchronisationspunkt derselben oder einer beliebigen Teilprozessabbild-ID erreicht wurde (`TimeSinceSameSyncPoint_ns` / `TimeSinceAnySyncPoint_ns`).

Die Funktion `RunToNextSyncPoint()` hebt den Freeze-Zustand auf.

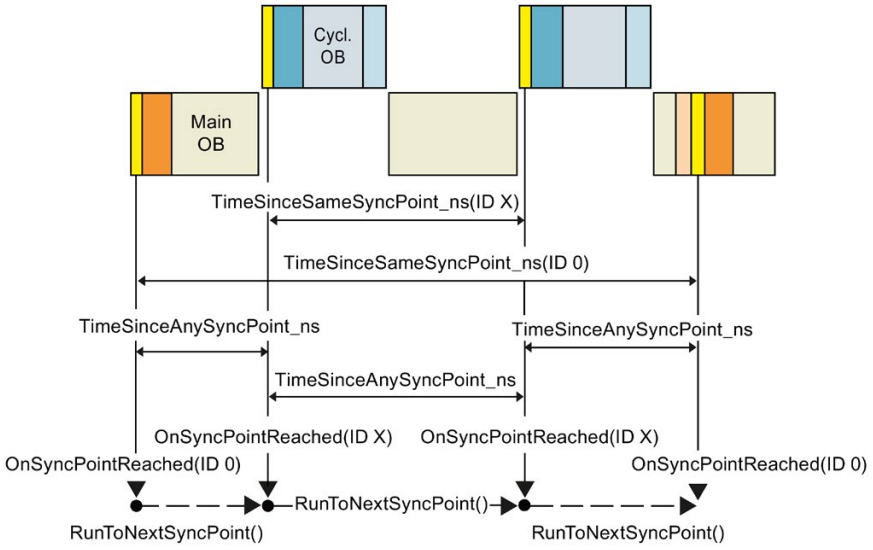


Bild 6-3 Beispiel: Ablauf in der Betriebsart SingleStep_CP

Einstellungen in der Beobachtungstabelle ändern

Hinweis

Trigger wählen für das Beobachten von Variablen in den SingleStep-Betriebsarten

Im TIA Portal zeigt die Beobachtungstabelle im Basismodus die Werte für Ausgänge und Merker **vor** der Verarbeitung.

Um die Variablenwerte **nach** der Verarbeitung anzuzeigen, wählen Sie für die Beobachtungstabelle den erweiterten Modus und wählen Sie dann in der Spalte "Beobachten mit Trigger" "Zyklusende, permanent".

Siehe auch

Zykluskontrolle (Seite 248)

6.3.2 Simulations-Partner zeitgesteuert synchronisieren

TimespanSynchronized-Betriebsarten

Mit den TimespanSynchronized-Betriebsarten des virtuellen Controllers werden mehrere Simulations-Partner (Clients) zeitgesteuert synchronisiert. Die Betriebsarten definieren den Synchronisationspunkt, an dem der virtuelle Controller in den Freeze-Zustand wechselt und das Ereignis `OnSyncPointReached` sendet.

Tabelle 6- 2 Zeitgesteuerte Betriebsarten (TimespanSynchronized)

Betriebsart	Synchronisationspunkt	
	Zykluskontrollpunkt	Vor dem Einlesen des Teilprozessabbilds
	"C"	"P"
`TimespanSynchronized_C`	✓	
`TimespanSynchronized_CP`	✓	✓
`TimespanSynchronized_P`		✓

API-Funktionen / Ereignisse

- Einstellungen zur Zykluskontrolle (Seite 248)
- StartProcessing() (Seite 254)
- OnSyncPointReached (Seite 288)

Freeze-Zustand beenden

Die Funktion `StartProcessing(t)` hebt den Freeze-Zustand auf und veranlasst den virtuellen Controller, mindestens so lange wie angefordert weiterzulaufen (auf Basis der virtuellen Zeit t), bevor er am nächsten Synchronisationspunkt wieder in den Freeze-Zustand geht.

Auch ein Wechsel in den Betriebszustand Default beendet den Freeze-Zustand.

Beispiel

Die Abbildung zeigt schematisch den Ablauf in der Betriebsart `TimespanSynchronized_CP`.

Neben dem Ereignis `OnSyncPointReached` sendet der virtuelle Controller auch die Laufzeit seit dem letzten Aufruf der Funktion `StartProcessing`(t) (`TimeSinceSameSyncPoint_ns` / `TimeSinceAnySyncPoint_ns`).

Die Funktion `StartProcessing()` hebt den Freeze-Zustand auf.

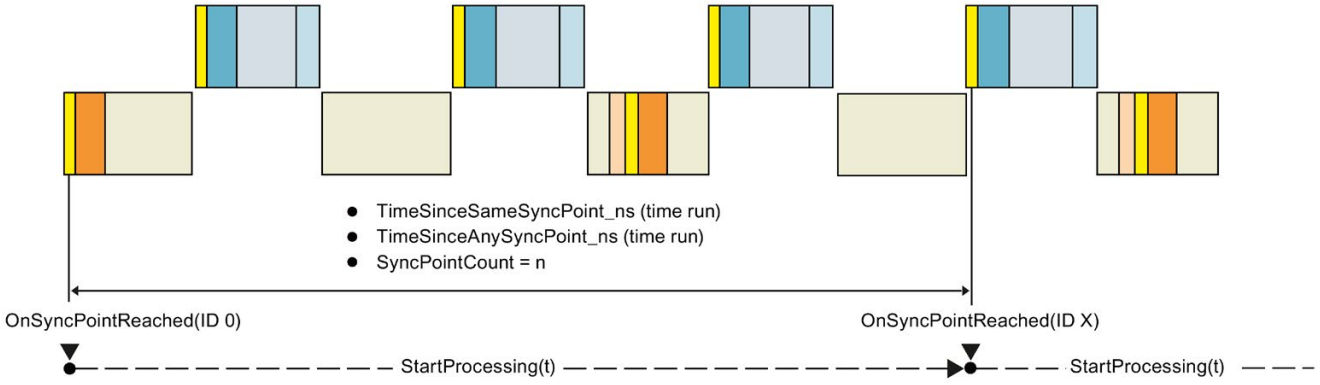


Bild 6-4 Beispiel: Ablauf in der Betriebsart TimespanSynchronized_CP

Beschreibung

Für die zeitgesteuerten Betriebsarten werden mindestens zwei Clients auf Basis einer virtuellen Zeitspanne synchronisiert. Ein Client kann eine Instanz eines virtuellen Controllers sein oder eine Anwendung, die die Runtime API nutzt (API-Client). Die Synchronisierung muss von einem Synchronisations-Master durchgeführt werden.

Der Synchronisations-Master beauftragt einen Client, eine bestimmte Zeitspanne zu laufen. Die Zeitspanne gibt der Master in Nanosekunden vor. Der Client läuft dann für die erwartete Zeitspanne, bevor er am nächsten Synchronisationspunkt in den Freeze-Zustand geht. Vor dem Wechsel in den Freeze-Zustand sendet der Client an den Master die genaue Zeitspanne, die er aktuell benötigt hat. Danach signalisiert der Master dem nächsten Client, aufzuholen.

API-Client als Master

Der API-Client als Master signalisiert jedem Client, wann er starten soll. Der Master erhält von jedem Client Ereignisse, wenn sie eingetreten sind.

Ein API-Client kann nur Instanzen eines virtuellen Controllers "zeitlich verwalten". Der API-Client erhält keine Ereignisse von anderen API-Clients. Er kann keine Meldungen an andere API-Clients senden.

Anwenderschnittstellen (API) 7

7.1 Einführung

Komponenten der Simulation Runtime

Für den Umgang mit der Simulation Runtime der PLCSIM Advanced sind folgende Komponenten relevant:

Tabelle 7- 1 Komponenten der Simulation Runtime

Komponenten	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. Er wird automatisch beendet, sobald keine Applikation mehr läuft, die 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.
• "SimulationRuntimeApi.h"	Header-Datei, die alle Datentypen beschreibt, die eine Native C++ Anwendung benötigt, um die API-Bibliothek zu verwenden.
• "Siemens.Simatic.PlcSim.Vplc1500. ODKClient.so.exe"	ODK-Client-Prozess für eine CPU Funktionsbibliothek (original Shared Object)
• "Siemens.Simatic.PlcSim.Vplc1500. ODKClient.x86.exe"	ODK-Client-Prozess für eine 32 Bit-Anwendung
• "Siemens.Simatic.PlcSim.Vplc1500. ODKClient.x64.exe"	ODK-Client-Prozess für eine 64 Bit-Anwendung

Externe Applikationen und Simulation Runtime

Die folgende Abbildung zeigt schematisch den Zugriff externer Applikationen über die Runtime API auf die Simulation Runtime. Der Simulation Runtime Manager verwaltet die Runtime Instanzen. Diese laden die Bibliotheken der virtuellen Controller.

Eine externe Applikation kann z. B. eine weitere Simulations-Software oder eine grafische Oberfläche (GUI) sein.

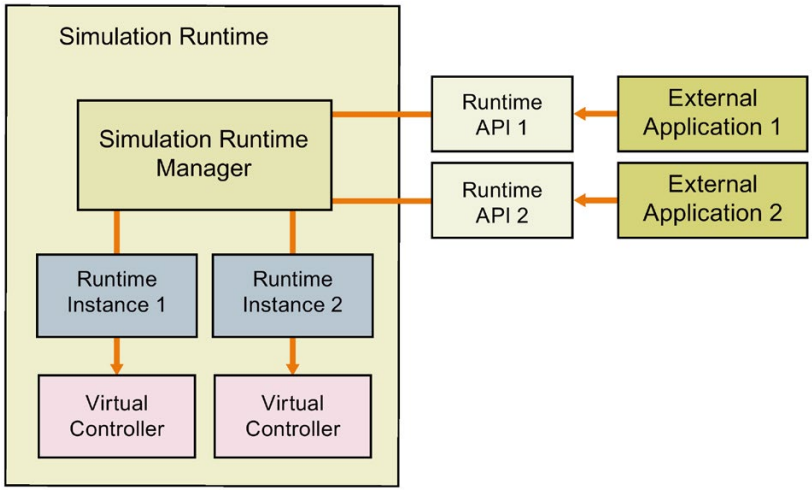
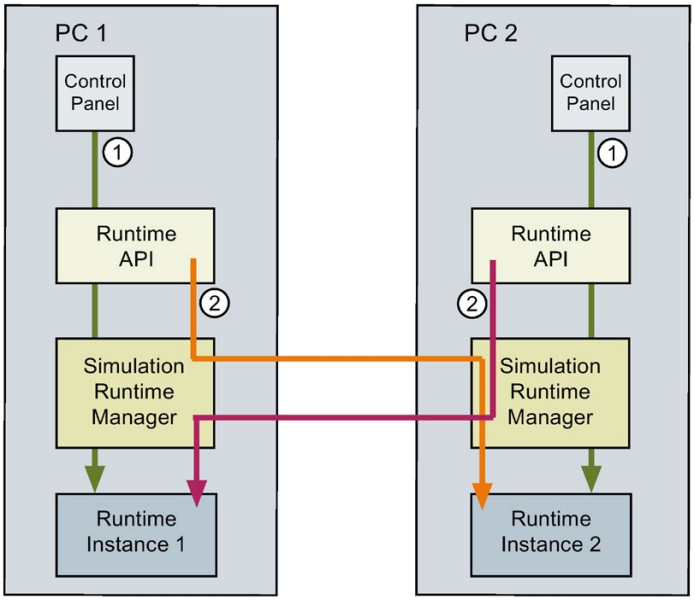


Bild 7-1 Externe Applikationen und Simulation Runtime

7.1.1 Zugriff auf Instanzen

Zugriff über das Control Panel und die API

Über das Control Panel können Sie nur auf eine Instanz zugreifen, die lokal auf dem PC vorhanden ist. Es spielt keine Rolle, auf welchem der PCs eine Instanz erstellt und gestartet wurde. Bei der verteilten Kommunikation greift die Runtime API über den Simulation Runtime Manager auf die Instanz des andern PCs zu.



① Zugriff auf eine lokale Instanz über das Control Panel
② Zugriff auf eine Remote-Instanz über die Runtime API

Bild 7-2 Zugriff auf Instanzen bei verteilter Kommunikation

API-Funktionen

- Tabelle 7-6 Übersicht IInstances Funktionen - Native C++ (Seite 103)
- Tabelle 7-13 Übersicht IInstances Funktionen - .NET (C#) (Seite 106)
- Tabelle 7-8 Übersicht IRemoteRuntimeManager Funktionen - Native C++ (Seite 105)
- Tabelle 7-15 Übersicht IRemoteRuntimeManager Funktionen - .NET (C#) (Seite 108)

Siehe auch

Übersicht Anwenderschnittstellen für Managed Code (Seite 105)

7.1.2 Anwenderschnittstellen (API)

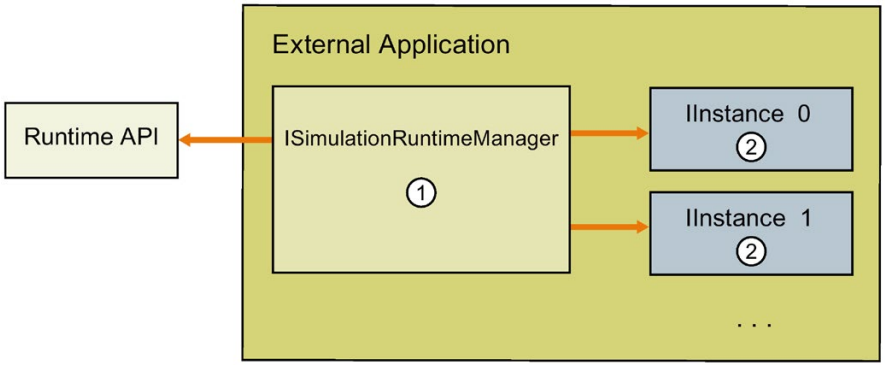
Die Anwenderschnittstellen der Simulation Runtime enthalten die Funktionen, mit denen Sie z. B. Instanzen erzeugen, den Betriebszustand eines virtuellen Controllers ändern und I/O-Daten austauschen.

Die Simulation Runtime enthält folgende Anwenderschnittstellen:

- ISimulationRuntimeManager
- IInstances
- IRemoteRuntimeManager

API und externe Anwendungen

Die Runtime API stellt einer externen Anwendung die Schnittstellen zur Verfügung.



① **ISimulationRuntimeManager**
Schnittstelle des Runtime Managers. Sie 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.

② **IInstances**
Schnittstelle einer Runtime Instanz. Sie wird benutzt, um den Betriebszustand eines virtuellen Controllers zu ändern und I/O-Daten auszutauschen. Jede Instanz hat einen eindeutigen Namen und eine ID.

Bild 7-3 API und externe Anwendungen

Zugriff auf API-Funktionen und Datentypen

Erforderliche Funktionen und Datentypen stehen für Native C++ und .NET (C#) zur Verfügung.

- Übersicht Anwenderschnittstellen für Native C++ (Seite 101)
- Übersicht Datentypen für Native C++ (Seite 109)
- Übersicht Anwenderschnittstellen für Managed Code (Seite 105)
- Übersicht Datentypen für Managed Code (Seite 111)

Hinweis

Einen direkten Zugriff auf die Beschreibung der einzelnen Funktionen und Datentypen erhalten Sie über das Tabellenverzeichnis in diesem Handbuch.

Übergabeparameter für API-Funktionen

Alle API-Funktionen, die einen Wert über die Funktionsparameter zurückliefern, erwarten als Übergabeparameter einen benutzerallokierten Speicherbereich. Null-Zeiger sind nicht zulässig. Eine Ausnahme hiervon sind die Funktionen, die eine Schnittstelle eines virtuellen Controllers zurückgeben:

- eine ISimulationRuntimeManager-Schnittstelle
- eine IRemoteRuntimeManager-Schnittstelle
- eine IInstance-Schnittstelle

7.1.3 Übersicht Anwenderschnittstellen für Native C++

API initialisieren und herunterfahren

Tabelle 7- 2 Übersicht API initialisieren und herunterfahren - Native C++

Aktionen	Funktionen
API initialisieren	InitializeApi (Seite 113) RuntimeApiEntry_Initialize (Seite 115)
API herunterfahren (Seite 117)	`DestroyInterface` `RuntimeApiEntry_DestroyInterface`
API-Bibliothek abmelden (Seite 120)	`FreeApi` `ShutdownAndFreeApi`

Globale Funktionen

Tabelle 7- 3 Übersicht Globale Funktionen - Native C++

Aktionen	Funktionen

Globale Funktionen (Seite 122)	`GetNameOfAreaSection()`
	`GetNameOfCPUType()`
	`GetNameOfCommunicationInterface()`
	`GetNameOfDataType()`
	`GetNameOfLEDMode()`
	`GetNameOfLEDType()`
	`GetNameOfOperatingMode()`
	`GetNameOfOperatingState()`
	`GetNameOfPrimitiveDataType()`
	`GetNameOfTagListDetails()`
	`GetNameOfErrorCode()`
	`GetNameOfRuntimeConfigChanged()`
	`GetNameOfInstanceConfigChanged()`
	`GetNameOfDiagSeverity()`
	`GetNameOfDirection()`
	`GetNameOfRackOrStationFaultType()`
	`GetNameOfProcessEventType()`
	`GetNameOfPullOrPlugEventType()`
	`GetNameOfCycleTimeMonitoringMode()`
	`GetNameOfDiagProperty()`
	`GetNameOfAutodiscoverType()`

API ISimulationRuntimeManager

Tabelle 7- 4 Übersicht API ISimulationRuntimeManager Funktionen - Native C++

Einstellungen	Funktionen

Schnittstelle (Seite 127)	`GetVersion()`
	`IsInitialized()`
	`IsRuntimeManagerAvailable()`
	`Shutdown()`

Simulation Runtime Instanzen (Seite 129)	`GetRegisteredInstancesCount()`
	`GetRegisteredInstanceInfoAt()`
	`RegisterInstance()`
	`RegisterCustomInstance()`
	`CreateInterface()`

Remote-Verbindungen (Seite 137)	`OpenPort()`
	`ClosePort()`
	`GetPort()`
	`GetRemoteConnectionsCount()`
	`GetRemoteConnectionInfoAt()`
	`RemoteConnect()`
	`RunAutodiscover()`

Tabelle 7- 5 Übersicht API ISimulationRuntimeManager Ereignisse - Native C++

Ereignisse	Funktionen

OnConfigurationChanged (Seite 144)	`RegisterOnConfigurationChangedCallback()`
	`UnregisterOnConfigurationChangedCallback()`
	`RegisterOnConfigurationChangedEvent()`
	`UnregisterOnConfigurationChangedEvent()`
	`WaitForOnConfigurationChangedEvent()`

OnRuntimeManagerLost (Seite 147)	`RegisterOnRuntimeManagerLostCallback()`
	`UnregisterOnRuntimeManagerLostCallback()`
	`RegisterOnRuntimeManagerLostEvent()`
	`UnregisterOnRuntimeManagerLostEvent()`
	`WaitForOnRuntimeManagerLostEvent()`

OnAutodiscover (Seite 150)	`RegisterOnAutodiscoverCallback()` `UnregisterOnAutodiscoverCallback()`

API IInstances

Tabelle 7- 6 Übersicht IInstances Funktionen - Native C++

Einstellungen	Funktionen

Schnittstelle (Seite 151)	`GetID()`
	`GetName()`
	`GetCPUType()`
	`SetCPUType()`
	`GetCommunicationInterface()`
	`SetCommunicationInterface()`
	`GetInfo()`
	`UnregisterInstance()`

Controller (Seite 157)	`GetControllerName()`
	`GetControllerShortDesignation()`
	`GetControllerIPCount()`
	`GetControllerIP()`
	`GetControllerIPSuite4()`
	`SetIPSuite()`
	`GetStoragePath()`
	`SetStoragePath()`
	`ArchiveStorage()`
	`RetrieveStorage()`
	`CleanupStoragePath()`

Betriebszustand (Seite 168)	`PowerOn()`
	`PowerOff()`
	`Run()`
	`Stop()`
	`GetOperatingState()`
	`MemoryReset()`

Variablentabelle (Seite 179)	`UpdateTagList()`
	`GetTagListStatus()`
	`GetTagInfoCount()`
	`GetTagInfos()`
	`CreateConfigurationFile()`

I/O-Zugriff über Adresse - Lesen (Seite 185)	`GetAreaSize()`
	`ReadBit()`
	`ReadByte()`
	`ReadBytes()`
	`ReadSignals()`

I/O-Zugriff über Adresse - Schreiben (Seite 194)	`WriteBit()`
	`WriteByte()`
	`WriteBytes()`
	`WriteSignals()`

Einstellungen	Funktionen

I/O-Zugriff über Variablenname - Lesen (Seite 201)	`Read()`
	`ReadBool()`
	`ReadChar(), ReadWChar()`
	`ReadDouble()`
	`ReadFloat()`
	`ReadInt8(), ReadInt16(), ReadInt32(), ReadInt64()`
	`ReadUInt8(), ReadUInt16(), ReadUInt32(), ReadUInt64()`
	`ReadSignals()`

I/O-Zugriff über Variablenname - Schreiben (Seite 224)	`Write()`
	`WriteBool()`
	`WriteChar(), WriteWChar()`
	`WriteDouble()`
	`WriteFloat()`
	`WriteInt8(), WriteInt16(), WriteInt32(), WriteInt64(),`
	`WriteUInt8(),WriteUInt16(), WriteUInt32(), WriteUInt64()`
	`WriteSignals()`

Virtuelle Zeit (Seite 245)	`GetSystemTime()`
	`SetSystemTime()`
	`GetScaleFactor()`
	`SetScaleFactor()`

Zykluskontrolle (Seite 248)	`GetOperatingMode()`
	`SetOperatingMode()`
	`SetSendSyncEventInDefaultModeEnabled()`
	`IsSendSyncEventInDefaultModeEnabled`
	`GetOverwrittenMinimalCycleTime_ns()`
	`SetOverwrittenMinimalCycleTime_ns()`
	`RunToNextSyncPoint()`
	`StartProcessing()`
	`SetCycleTimeMonitoringMode()`
	`GetCycleTimeMonitoringMode()`

Azyklische Dienste (Seite 258)	

Tabelle 7- 7 Übersicht IInstances Ereignisse - Native C++

Ereignisse	Funktionen

OnOperatingStateChanged (Seite 278)	`RegisterOnOperatingStateChangedCallback()`
	`UnregisterOnOperatingStateChangedCallback()`
	`RegisterOnOperatingStateChangedEvent()`
	`UnregisterOnOperatingStateChangedEvent()`
	`WaitForOnOperatingStateChangedEvent()`

OnLedChanged (Seite 281)	`RegisterOnLedChangedCallback()`
	`UnregisterOnLedChangedCallback()`
	`RegisterOnLedChangedEvent()`
	`UnregisterOnLedChangedEvent()`
	`WaitForOnLedChangedEvent()`

OnConfigurationChanging (Seite 283)	`RegisterOnConfigurationChangingCallback()`
	`UnregisterOnConfigurationChangingCallback()`
	`RegisterOnConfigurationChangingEvent()`
	`UnregisterOnConfigurationChangingEvent()`
	`WaitForOnConfigurationChangingEvent()`

OnConfigurationChanged (Seite 286)	`RegisterOnConfigurationChangedCallback()`
	`UnregisterOnConfigurationChangedCallback()`
	`RegisterOnConfigurationChangedEvent()`
	`UnregisterOnConfigurationChangedEvent()`
	`WaitForOnConfigurationChangedEvent()`

OnSyncPointReached (Seite 288)	`RegisterOnSyncPointReachedCallback()UnregisterOnSyncPointReachedCallback()RegisterOnSyncPointReachedEvent()UnregisterOnSyncPointReachedEvent()WaitForOnSyncPointReachedEvent()`

API IRemoteRuntimeManager

Tabelle 7- 8 Übersicht IRemoteRuntimeManager Funktionen - Native C++

Einstellungen	Funktionen

Schnittstelle (Seite 301)	`GetVersion()`
	`GetIP()`
	`GetPort()`
	`GetRemoteComputerName()`
	`Disconnect()`

Simulation Runtime Instanzen (Seite 305)	`GetRegisteredInstancesCount()`
	`GetRegisteredInstanceInfoAt()`
	`RegisterInstance()`
	`RegisterCustomInstance()`
	`CreateInterface()`

Tabelle 7- 9 Übersicht IRemoteRuntimeManager Ereignisse - Native C++

Ereignisse	Funktionen

OnConnectionLost (Seite 313)	`RegisterOnConnectionLostCallback()`
	`UnregisterOnConnectionLostCallback()`
	`RegisterOnConnectionLostEvent()`
	`UnregisterOnConnectionLostEvent()`
	`WaitForOnConnectionLostEvent()`

7.1.4 Übersicht Anwenderschnittstellen für Managed Code

API initialisieren und herunterfahren

Tabelle 7- 10 Übersicht API initialisieren und herunterfahren - .NET (C#)

Aktionen	Funktionen
API initialisieren (Seite 117)	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeManager`
API herunterfahren (Seite 122)	

API ISimulationRuntimeManager

Tabelle 7- 11 Übersicht ISimulationRuntimeManager Funktionen - .NET (C#)

Einstellungen	Funktionen

Schnittstelle (Seite 127)	`Version { get; }`
	`IsInitialized { get; }`
	`IsRuntimeManagerAvailable { get; }`
	`Shutdown()`

Simulation Runtime Instanzen (Seite 129)	`RegisterInstanceInfo { get; }`
	`RegisterInstance()`
	`RegisterCustomInstance()`
	`CreateInterface()`

Remote Verbindungen (Seite 137)	`OpenPort()`
	`ClosePort()`
	`Port { get; }`
	`RemoteConnectionInfo { get; }`
	`RemoteConnect()`
	`RunAutodiscover()`

Tabelle 7- 12 Übersicht ISimulationRuntimeManager Ereignisse - .NET (C#)

Ereignisse	Funktionen

OnConfigurationChanged (Seite 144)	`OnConfigurationChanged`
	`RegisterOnConfigurationChangedEvent()`
	`UnregisterOnConfigurationChangedEvent()`
	`WaitForOnConfigurationChangedEvent()`

OnRuntimeManagerLost (Seite 147)	`OnRuntimeManagerLost()`
	`RegisterOnRuntimeManagerLostEvent()`
	`UnregisterOnRuntimeManagerLostEvent()`
	`WaitForOnRuntimeManagerLostEvent()`

OnAutodiscover (Seite 150)	`OnAutodiscoverData`

API IInstances

Tabelle 7- 13 Übersicht IInstances Funktionen - .NET (C#)

Einstellungen	Funktionen

Schnittstelle (Seite 151)	`Dispose ()`
	`ID { get; }`
	`Name { get; }`
	`CPUType { get; set; }`
	`CommunicationInterface { get; }`
	`Info { get; }`
	`UnregisterInstance()`

Controller - Informationen und Einstellungen (Seite 157)	`ControllerName { get; }`
	`ControllerShortDesignation { get; }`
	`ControllerIPSuite4 { get; }`
	`SetIPSuite()`
	`StoragePath { get; set; }`
	`ArchiveStorage()`
	`RetrieveStorage()`
	`CleanupStoragePath()`

Einstellungen	Funktionen

Betriebszustand (Seite 168)	`PowerOn()`
	`PowerOff()`
	`Run()`
	`Stop()`
	`OperatingState { get; }`
	`MemoryReset()`

Variablentabelle (Seite 179)	`UpdateTagList()`
	`GetTagListStatus()`
	`TagInfos { get; }`
	`CreateConfigurationFile()`

I/O-Zugriff über Adresse - Lesen (Seite 185)	`InputArea | MarkerArea | OutputArea { get; }`
	`AreaSize { get; }`
	`ReadBit()`
	`ReadByte()`
	`ReadBytes()`
	`ReadSignals()`

I/O-Zugriff über Adresse - Schreiben (Seite 194)	`WriteBit()`
	`WriteByte()`
	`WriteBytes()`
	`WriteSignals()`

I/O-Zugriff über Variablenname - Lesen (Seite 201)	`Read()`
	`ReadBool()`
	`ReadChar(), ReadWChar()`
	`ReadDouble()`
	`ReadFloat()`
	`ReadInt8(), ReadInt16(), ReadInt32(), ReadInt64()`
	`ReadUInt8(), ReadUInt16(), ReadUInt32(), ReadUInt64()`
	`ReadSignals()`

I/O-Zugriff über Variablenname - Schreiben (Seite 224)	`Write()`
	`WriteBool()`
	`WriteChar(), WriteWChar()`
	`WriteDouble()`
	`WriteFloat()`
	`WriteInt8(), WriteInt16(), WriteInt32(), WriteInt64(),`
	`WriteUInt8(),WriteUInt16(), WriteUInt32(), WriteUInt64()`
	`WriteSignals()`

Virtuelle Zeit (Seite 245)	`SystemTime { get; set; }` `ScaleFactor { get; set; }`

Zykluskontrolle (Seite 248)	`OperatingMode { get; set; }`
	`IsSendSyncEventInDefaultModeEnabled { get; set; }`
	`OverwrittenMinimalCycleTime_ns { get; set; }`
	`RunToNextSyncPoint`
	`StartProcessing()`
	`SetCycleTimeMonitoringMode()`
	`GetCycleTimeMonitoringMode()`

Azyklische Dienste (Seite 258)	

Tabelle 7- 14 Übersicht IInstances Ereignisse - .NET (C#)

Ereignisse	Funktionen

OnOperatingStateChanged (Seite 278)	`OnOperatingStateChanged`
	`RegisterOnOperatingStateChangedEvent()`
	`UnregisterOnOperatingStateChangedEvent()`
	`WaitForOnOperatingStateChangedEvent()`

OnLedChanged (Seite 281)	`OnLedChanged`
	`RegisterOnLedChangedEvent()`
	`UnregisterOnLedChangedEvent()`
	`WaitForOnLedChangedEvent()`

Ereignisse	Funktionen

OnConfigurationChanging (Seite 283)	`OnConfigurationChanging`
	`RegisterOnConfigurationChangingEvent()`
	`UnregisterOnConfigurationChangingEvent()`
	`WaitForOnConfigurationChangingEvent()`

OnConfigurationChanged (Seite 286)	`OnConfigurationChanged`
	`RegisterOnConfigurationChangedEvent()`
	`UnregisterOnConfigurationChangedEvent()`
	`WaitForOnConfigurationChangedEvent()`

OnSyncPointReached (Seite 288)	`OnSyncPointReached`
	`RegisterOnSyncPointReachedEvent()`
	`UnregisterOnSyncPointReachedEvent()`
	`WaitForOnSyncPointReachedEvent()`

API IRemoteRuntimeManager

Tabelle 7- 15 Übersicht IRemoteRuntimeManager Funktionen - .NET (C#)

Einstellungen	Funktionen

Schnittstelle (Seite 301)	`Dispose()`
	`Version { get; }`
	`IP { get; }`
	`Port { get; }`
	`RemoteComputerName { get; }`
	`Disconnect()`

Simulation Runtime Instanzen (Seite 129)	`RegisterInstanceInfo { get; }`
	`RegisterInstance()`
	`RegisterCustomInstance()`
	`CreateInterface()`

Tabelle 7- 16 Übersicht IRemoteRuntimeManager Ereignisse - .NET (C#)

Ereignisse	Funktionen

OnConnectionLost() (Seite 313)	`OnConnectionLost()`
	`RegisterOnConnectionLostEvent()`
	`UnregisterOnConnectionLostEvent()`
	`WaitForOnConnectionLostEvent()`

7.1.5 Übersicht Datentypen für Native C++

Die folgende Tabelle zeigt, welche Datentypen für die Simulation im Runtime Manager vorhanden sind.

Tabelle 7- 17 Übersicht Datentypen - Native C++

Datentyp	
DLL-Importfunktionen (Seite 317)	`ApiEntry_Initialize` `ApiEntry_DestroyInterface`

Event Callback- Funktionen (Seite 318)	`EventCallback_VOID`
	`EventCallback_SRCC_UINT32_UINT32_INT32`
	`EventCallback_SRRSI_AD`
	`EventCallback_IRRTM`
	`EventCallback_II_SREC_ST_SROS_SROS`
	`EventCallback_II_SREC_ST_UINT32_INT64_INT64_UINT32`
	`EventCallback_II_SREC_ST`
	`EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32`
	`EventCallback_II_SREC_ST_SRLT_SRLM`
	`EventCallback_II_SREC_ST_SDRI`
	`EventCallback_II_SREC_ST_SDRI_BYTE`
	`EventCallback_II_SREC_ST_UINT32_UINT32`
	`EventCallback_II_SREC_ST_UINT32_UINT32_EPET_UINT32`
	`EventCallback_II_SREC_ST_UINT32_EPPET_UINT32`
	`EventCallback_II_SREC_ST_UINT32_ERSFET`
	`EventCallback_II_SREC_ST_UINT32`

Definitionen und Konstanten (Seite 344)	
Unions (Seite 346)	`UIP` `UDataValue`

Datentyp	
Strukturen (Seite 348)	SDataValue SDVBNI SDataValueByAddress SDataValueByName SConnectionInfo SInstanceInfo SDimension STagInfo SIP SIPSuite4 SOnSyncPointReachedResult SDataValueByAddressWithCheck SDataValueByNameWithCheck SDataRecordInfo SDataRecord SConfiguredProcessEvents SDiagExtChannelDescription SAutodiscoverData
Aufzählungen (Seite 370)	ERuntimeErrorCode EArea EOperatingState EOperatingMode ECPUType ECommunicationInterface ELEDType ELEDMode EPrimitiveDataTypes EDataType ETagListDetails ERuntimeConfigChanged EInstanceConfigChanged EPullOrPlugEventType EProcessEventType EDirection EDiagProperty EDiagSeverity ERackOrStationFaultType ECycleTimeMonitoringMode EAutodiscoverType

7.1.6 Übersicht Datentypen für Managed Code

Die folgende Tabelle zeigt, welche Datentypen für die Simulation im Runtime Manager vorhanden sind.

Tabelle 7- 18 Übersicht Datentypen - .NET (C#)

Datentyp	
Delegat Definitionen (Seite 331) - Event Handler Methoden	Delegate_Void Delegate_SRCC_UINT32_UINT32_INT32 Delegate_SRRSI_AD Delegate_IRRTM Delegate_II_EREC_DT_EOS_EOS Delegate_II_EREC_DT_UINT32_INT64_INT64_UINT32 Delegate_II_EREC_DT Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32 Delegate_II_EREC_DT_ELT_ELM Delegate_II_EREC_DT_SDRI Delegate_II_EREC_DT_SDR Delegate_SREC_ST_UINT32_UINT32 Delegate_SREC_ST_UINT32_UINT32_EPET_UINT32 Delegate_SREC_ST_UINT32_EPPET_UINT32 Delegate_SREC_ST_UINT32_ERSFET Delegate_SREC_ST_UINT32
Definitionen und Konstanten (Seite 344)	
Strukturen (Seite 348)	SDataValue SDVBNI SDataValueByAddress SDataValueByName SConnectionInfo SInstanceInfo SDimension STagInfo SIP SIPSuite4 SOnSyncPointReachedResult SDataValueByAddressWithCheck SDataValueByNameWithCheck SDataRecordInfo SDataRecord SConfiguredProcessEvents SDiagExtChannelDescription SAutodiscoverData
Aufzählungen (Seite 369)	ERuntimeErrorCode EArea EOperatingState EOperatingMode ECPUType ECommunicationInterface ELEDType ELEDMode EPrimitiveDataTypes EDataType ETagListDetails ERuntimeConfigChanged EInstanceConfigChanged EPullOrPlugEventType EProcessEventType EDirection EDiagProperty EDiagSeverity ERackOrStationFaultType ECycleTimeMonitoringMode EAutodiscoverType

7.2 API initialisieren

7.2.1 API-Bibliothek laden

Beschreibung

Bei PLCSIM Advanced sind die Schnittstellen der API V3.0 nicht kompatibel mit den Schnittstellen früherer API Versionen. Der Runtime Manager von PLCSIM Advanced V3.0 ist aber kompatibel mit der API von früheren PLCSIM Advanced Versionen.

Beim Installieren von PLCSIM Advanced V3.0 werden auch frühere Versionen der API installiert.

Der voreingestellte Pfad lautet:

- C:\Program Files (x86)\Common Files\Siemens\PLCSIMADV\API\1.0
- C:\Program Files (x86)\Common Files\Siemens\PLCSIMADV\API\2.0
- C:\Program Files (x86)\Common Files\Siemens\PLCSIMADV\API\2.1
- C:\Program Files (x86)\Common Files\Siemens\PLCSIMADV\API\3.0

Den Installationspfad von PLCSIM Advanced entnehmen Sie der Registry:

- Schlüssel: "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Siemens\Shared Tools\PLCSIMADV_SimRT"
- Wert: "Path"

Um den Pfad zur API zu erhalten, fügen Sie an das Ende der Zeichenfolge folgendes Unterverzeichnis hinzu: "API\<API version>" (z. B. "API\3.0").

Wenn Sie diesen Pfad nutzen, dann wird die API-Bibliothek (DLL) direkt vom Installationsverzeichnis geladen.

Verweis

Weitere Informationen erhalten Sie:

- Für Native C++ im Kapitel InitializeApi() (Seite 113).
- Für .NET über den Aufruf der Funktion "System.Reflection.Assembly.LoadFile(string)" in der Online-Dokumentation zu MSDN.

7.2.2 Native C++

7.2.2.1 InitializeApi()

Beschreibung

Die Funktion `InitializeApi` lädt die API-Bibliothek (DLL) und initialisiert die API. Die Funktion lädt die Version der DLL, die zur Architektur Ihrer Anwendung passt und die auch mit der Header-Datei der API kompatibel ist ("SimulationRuntimeApi.h").

Um die DLL zu laden, sucht die Funktion `InitializeApi` der Reihe nach in folgenden Verzeichnissen:

- Im Verzeichnis, zu dem der Parameter der Funktion führt (`in_SimulationRuntimeApiDllPath`)
- Im Verzeichnis, in dem auch Ihre Anwendung liegt, die diese Funktion aufruft
- Im Installationsverzeichnis von PLCSIM Advanced

Wenn keine DLL vorliegt, greift die Funktion auf das nächste Verzeichnis zu.

Die Funktion liefert an den Simulation Runtime Manager eine Schnittstelle zurück. Nutzen Sie diese Schnittstelle, um eine neue Instanz des virtuellen Controllers zu erzeugen oder um Zugriff auf eine bereits bestehende Instanz zu erhalten.

Tabelle 7- 19 InitializeApi() - Native C++

Syntax	`ERuntimeErrorCode InitializeApi(`
	`  ISimulationRuntimeManager** out_SimulationRuntimeManagerInterface`
	`);`
	`ERuntimeErrorCode InitializeApi(`
	`  WCHAR* in_SimulationRuntimeApiDllPath,`
	`  ISimulationRuntimeManager** inout_SimulationRuntimeManagerInter-`
	`face`
	`);`

Parameter	• `ISimulationRuntimeManager** out_SimulationRuntimeManagerInter-face:` Zeiger auf einen Runtime Manager Schnittstellenzeiger. Der Zeiger muss mit `NULL` initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt. Siehe Datentypen (Seite 316). • `WCHAR* in_SimulationRuntimeApiDllPath`: Der Pfad zur Runtime API-Bibliothek.

Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WRONG_ARGUMENT`	Der Zeiger auf die Runtime Manager Schnittstelle ist ungleich `NULL`.
	`SREC_WRONG_VERSION`	• Die angeforderte Version der Schnittstelle ist nicht kompatibel mit der Version, mit der die API kompiliert wurde. • Die Version der API ist nicht kompatibel mit der Runtime. Siehe Kompatibilität beim Upgrade (Seite 33).
	`SREC_CONNECTION_ERROR`	Zum Runtime Manager kann keine Verbindung hergestellt werden.
	`SREC_ERROR_LOADING_DLL`	Die API-Bibliothek kann nicht geladen werden.
	`SREC_RUNTIME_NOT_AVAILABLE`	In dieser Windows-Nutzersitzung läuft kein Runtime Manager.

Beispiel C++	`// Include The Headerfile Of The API`
	`#include "SimulationRuntimeApi.h"`
	
	`// Prepare The Variables`
	`ERuntimeErrorCode result = SREC_INVALID_ERROR_CODE;`
	`ISimulationRuntimeManager* api = NULL;`
	
	`// Initialize The API And Get The RuntimeManager Interface`
	`result = InitializeApi(&api);`	

Hinweis

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118).

7.2.2.2 RuntimeApiEntry_Initialize

Beschreibung

Nutzen Sie die Funktion `RuntimeApiEntry_Initialize` nur, wenn die API-Bibliothek (DLL) von einem anderen Verzeichnis geladen werden soll als dem Verzeichnis, in dem auch Ihre Anwendung liegt, die diese Funktion aufruft.

Beim Initialisieren der API wird zuerst die API-Bibliothek geladen und dann die Funktion `Initialize` importiert und aufgerufen.

Die Funktion liefert an den Simulation Runtime Manager eine Schnittstelle zurück. Nutzen Sie diese Schnittstelle, um eine neue Instanz des virtuellen Controllers zu erzeugen oder um Zugriff auf eine bereits bestehende Instanz zu erhalten.

Tabelle 7- 20 RuntimeApiEntry_Initialize - Native C++

Syntax	`__declspec(dllexport) ERuntimeErrorCode RuntimeApiEntry_Initialize( ISimulationRuntimeManager** out_SimulationRuntimeManagerInterface, UINT32 in_InterfaceVersion );`	
Parameter	• `ISimulationRuntimeManager** out_SimulationRuntimeManagerInterface:` Zeiger auf einen Runtime Manager Schnittstellenzeiger. Der Zeiger muss mit `NULL` initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt. Siehe Datentypen (Seite 316). • `UINT32 in_InterfaceVersion:` Die Version der API-Schnittstelle, die geladen werden soll: `DAPI_DLL_INTERFACE_VERSION`.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WRONG_ARGUMENT`	Der Zeiger auf die Runtime Manager Schnittstelle ist ungleich `NULL`.
	`SREC_WRONG_VERSION`	• Die angeforderte Version der Schnittstelle ist nicht kompatibel mit der Version, mit der die API kompiliert wurde. • Die Version der API ist nicht kompatibel mit der Runtime. Siehe Kompatibilität beim Upgrade (Seite 33).
	`SREC_CONNECTION_ERROR`	Zum Runtime Manager kann keine Verbindung hergestellt werden.
	`SREC_RUNTIME_NOT_AVAILABLE`	In dieser Windows-Nutzersitzung läuft kein Runtime Manager.

Beispiel C++	

```
// Include The Headerfile Of The API
#include "SimulationRuntimeApi.h"


// Prepare The Variables
ERuntimeErrorCode result = SREC_INVALID_ERROR_CODE;
HMODULE dllHandle = NULL;
ApiEntry_Initialize Initialize = NULL;
ISimulationRuntimeManager* api = NULL;


// Load The DLL And Import The "Initialize" Function (using the Win32 API)
dllHandle = LoadLibrary(DAPI_DLL_NAME_X86);
if ( dllHandle != NULL )
{
 Initialize = (ApiEntry_Initialize)GetProcAddress(dllHandle, DAPI_ENTRY_INITIALIZE);
}


// Initialize The API And Get The RuntimeManager Interface
if ( Initialize != NULL )
{
 result = Initialize(&api, DAPI_DLL_INTERFACE_VERSION);
}
```

Hinweis

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118).

7.2.3 .NET (C#)

7.2.3.1 Initialize

Beschreibung

Der Eintrittspunkt zur API ist die statische Klasse
`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeManager`.
Die API wird initialisiert, wenn eine Funktion dieser Klasse erstmalig genutzt wird.

Tabelle 7- 21 Initialize - .NET (C#)

Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationInitializationException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.ConnectionError`	Zum Runtime Manager kann keine Verbindung hergestellt werden.
	`ERuntimeErrorCode.WrongVersion`	Die Version der API ist nicht kompatibel mit der Runtime. Siehe Kompatibilitat beim Upgrade (Seite 33).
	`ERuntimeErrorCode.RuntimeNotAvailable`	In dieser Windows Nutzersitzung läuft kein Runtime Manager.

7.3 API herunterfahren

7.3.1 Native C++

Prinzipielles Vorgehen beim Löschen der Anwenderschnittstellen

Um alle Anwenderschnittstellen zu löschen, gehen Sie prinzipiell so vor:

1. Löschen Sie die Schnittstellen IInstances und IRemoteRuntimeManager.
2. Rufen Sie die Funktion `Shutdown()` der Schnittstelle ISimulationRuntimeManager auf.
3. Löschen Sie die Schnittstelle ISimulationRuntimeManager.
4. Entladen Sie die API-Bibliothek (DLL) mit der Win32 API-Funktion `FreeLibrary()`.

Löschen der Anwenderschnittstellen über Funktionen

Das Löschen der Anwenderschnittstellen ist auch über Funktionen möglich.

Wenn die API über die Funktion `InitializeApi()` initialisiert wurde, dann löschen Sie die Anwenderschnittstellen über folgende Funktionen:

- FreeApi() (Seite 120)
- ShutdownAndFreeApi() (Seite 121)

7.3.1.1 DestroyInterface()

Beschreibung

Ein Funktionszeiger auf die Funktion `RuntimeApiEntry_DestroyInterface`. Der Funktionszeiger `DestoyInterface()` wird nur gültig, wenn die Funktion `InitializeApi` erfolgreich aufgerufen wurde.

Die Funktion entlädt den Speicher einer ISimulationRuntimeManager, IRemoteRuntimeManager oder IInstance Schnittstelle.

Tabelle 7- 22 DestroyInterface() - Native C++

Syntax	`ERuntimeErrorCode DestroyInterface( IBaseInterface* in_Interface );`	
Parameter	• `IBaseInterface* in_Interface:` Die Schnittstelle, die gelöscht werden soll.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WRONG_ARGUMENT`	Der Zeiger auf die Schnittstelle ist `NULL`.

Beispiel C++	`// Include The Headerfile Of The API`
	`#include "SimulationRuntimeApi.h"`
	
	`// The Interfaces`
	`ERuntimeErrorCode result;`
	`ISimulationRuntimeManager* api = NULL;`
	`IInstance* instance = NULL;`
	
	`// Init the DLL and create an instance`
	`result = InitializeApi(&api);`
	`result = api->RegisterInstance(&instance);`
	
	`// Destroy Instance Interfaces`
	`result = DestroyInterface(instance);`
	`instance = NULL;`	

7.3.1.2 RuntimeApiEntry_DestroyInterface

Beschreibung

Nutzen Sie die Funktion `RuntimeApiEntry_DestroyInterface` nur, wenn die API-Bibliothek (DLL) von einem anderen Verzeichnis geladen werden soll als dem Startup-Verzeichnis der Anwendung, die diese Funktion aufruft.

Wenn die API über die Funktion `InitializeApi` initialisiert wurde, dann wählen Sie die Funktion DestroyInterface() (Seite 118).

Die Funktion entlädt den Speicher einer ISimulationRuntimeManager, IRemoteRuntimeManager oder IInstance Schnittstelle.

Tabelle 7- 23 RuntimeApiEntry_DestroyInterface() - Native C++

Syntax	`__declspec(dllexport) ERuntimeErrorCode RuntimeApiEntry_DestroyInterface(`
	`IBaseInterface* in_Interface`
	`);`	

Parameter	• `IBaseInterface* in_Interface:` Die Schnittstelle, die gelöscht werden soll.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WRONG_ARGUMENT`	Der Zeiger auf die Schnittstelle ist `NULL`.

Beispiel C++	`// Include The Headerfile Of The API`
	`#include "SimulationRuntimeApi.h"`
	
	`// Prepare The Variables`
	`ERuntimeErrorCode result = SREC_INVALID_ERROR_CODE;`
	`HMODULE dllHandle = NULL;`
	`ApiEntry_DestroyInterface Destroy = NULL;`
	`IInstance* instance = NULL;`
	
	`// Load The DLL And Import The "DestroyInterface" Function (using the Win32 API)`
	`dllHandle = LoadLibraryA(DAPI_DLL_NAME_X86);`
	`if ( dllHandle != NULL )`
	`{`
	`Destroy = (ApiEntry_DestroyInterface)GetProcAddress(dllHandle, DAPI_ENTRY_DESTROY_INTERFACE);`
	`}`
	`…`
	`// Frees the memory of an IInstance interface`
	`result = Destroy(instance);`	

7.3.1.3 FreeApi()

Beschreibung

Die Funktion `FreeApi()` entlädt die Bibliothek der Runtime API.

Diese Funktion kann nur aufgerufen werden nach dem erfolgreichen Aufruf der Funktion `InitializeApi`. Wenn die Funktion `InitializeApi` nicht aufgerufen wurde, muss die Bibliothek über die Win32 API-Funktion `FreeLibrary()` entladen werden.

Tabelle 7- 24 FreeApi() - Native C++

Syntax	`ERuntimeErrorCode FreeApi();`	
Parameter	Keine	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_API_NOT_INITIALIZED`	Die Funktion `InitializeApi` wurde nicht erfolgreich aufgerufen.

Beispiel C++	`// Include The Headerfile Of The API`
	`#include "SimulationRuntimeApi.h"`
	
	`// The Interfaces`
	`ERuntimeErrorCode result;`
	`ISimulationRuntimeManager* api = NULL;`
	`IInstance* instance = NULL;`
	
	`// Init the API`
	`result = InitializeApi(&api);`
	
	`…`
	
	`// Shutdown The API`
	`api->Shutdown();`
	`result = DestroyInterface(api);`
	`api = NULL;`
	`result = FreeApi();`	

7.3.1.4 ShutdownAndFreeApi()

Beschreibung

Die Funktion `ShutdownAndFreeApi()` fährt die Runtime API herunter, löscht die IRuntimeManager Schnittstelle und entlädt die Bibliothek der Runtime API.

Diese Funktion kann nur aufgerufen werden nach dem erfolgreichen Aufruf der Funktion `InitializeApi`. Wenn die Funktion `InitializeApi` nicht aufgerufen wurde, muss die Bibliothek über die Win32 API-Funktion `FreeLibrary()` entladen werden.

Tabelle 7- 25 ShutdownAndFreeApi() - Native C++

Syntax	`ERuntimeErrorCode ShutdownAndFreeApi( ISimulationRuntimeManager* in_SimulationRuntimeManagerInterface );`	
Parameter	• `ISimulationRuntimeManager* in_SimulationRuntimeManagerInterface:` Die Schnittstelle des Runtime Managers, die gelöscht werden soll.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_API_NOT_INITIALIZED`	Die Funktion `InitializeApi` wurde nicht erfolgreich aufgerufen.
	`SREC_WRONG_ARGUMENT`	Der Zeiger auf die Schnittstelle ist `NULL`.

Beispiel C++	`// Include The Headerfile Of The API`
	`#include "SimulationRuntimeApi.h"`
	
	`// The Interfaces`
	`ERuntimeErrorCode result;`
	`ISimulationRuntimeManager* api = NULL;`
	`IInstance* instance = NULL;`
	
	`// Init the API`
	`result = InitializeApi(&api);`
	
	…
	
	`// Shutdown The API`
	`result = ShutdownAndFreeApi(api);`
	`api = NULL;`	

7.3.2 .NET (C#)

7.3.2.1 API herunterfahren

Die .NET-Komponenten der API können Sie für die IInstance und IRemoteRuntimeManager Schnittstellen über den Aufruf der Funktion Dispose (Seite 151) beenden.

Zudem können diese Schnittstellen auch automatisch durch den .NET Garbage Collector bereinigt werden.

API manuell bereinigen

Um die API manuell zu bereinigen, gehen Sie so vor:

1. Löschen Sie alle Schnittstellen. Schnittstellen - Informationen und Einstellungen (Seite 151)
2. Rufen Sie die Funktion Shutdown() (Seite 127) der ISimulationRuntimeManager Schnittstelle auf.

7.4 Globale Funktionen (Native C++)

Die globalen Funktionen `GetNameOf...` liefern den Namen des Aufzählungseintrags zurück (`const WCHAR*`).

GetNameOfAreaSection()

Tabelle 7- 26 GetNameOfAreaSection() - Native C++

Syntax	`const WCHAR* GetNameOfAreaSection(`
	` EArea in_AreaSection`
	`);`

Parameter	`EArea in_AreaSection:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfCPUType()

Tabelle 7- 27 GetNameOfCPUType() - Native C++

Syntax	`const WCHAR* GetNameOfCPUType(`
	` ECPUType in_CPUType`
	`);`

Parameter	`ECPUType in_CPUType:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfCommunicationInterface()

Tabelle 7- 28 GetNameOfCommunicationInterface() - Native C++

Syntax	`const WCHAR* GetNameOfCommunicationInterface( ECommunicationInterface in_CommunicationInterface );`
Parameter	`ECommunicationInterface in_CommunicationInterface`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfDataType()

Tabelle 7- 29 GetNameOfDataType() - Native C++

Syntax	`const WCHAR* GetNameOfDataType( EDataType in_DataType );`
Parameter	`EDataType in_DataType`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfErrorCode()

Tabelle 7- 30 GetNameOfErrorCode() - Native C++

Syntax	`const WCHAR* GetNameOfErrorCode( ERuntimeErrorCode in_ErrorCode );`
Parameter	`ERuntimeErrorCode in_ErrorCode`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfLEDMode()

Tabelle 7- 31 GetNameOfLEDMode() - Native C++

Syntax	`const WCHAR* GetNameOfLEDMode( ELEDMode in_LEDMode );`
Parameter	`ELEDMode in_LEDMode`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfLEDType()

Tabelle 7- 32 GetNameOfLEDType() - Native C++

Syntax	`const WCHAR* GetNameOfLEDType( ELEDType in_LEDType );`
Parameter	`ELEDType in_LEDType`: Aufzählungseintrag.
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfOperatingMode()

Tabelle 7- 33 GetNameOfOperatingMode() - Native C++

Syntax	`const WCHAR* GetNameOfOperatingMode(`
	`  EOperatingMode in_OperatingMode`
	`);`

Parameter	`EOperatingMode in_OperatingMode`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfErrorCode()

Tabelle 7- 34 GetNameOfErrorCode() - Native C++

Syntax	`const WCHAR* GetNameOfErrorCode(`
	`  ERuntimeErrorCode in_ErrorCode`
	`);`

Parameter	`ERuntimeErrorCode in_ErrorCode`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfOperatingState

Tabelle 7- 35 GetNameOfOperatingState() - Native C++

Syntax	`const WCHAR* GetNameOfOperatingState(`
	`  EOperatingState in_OperatingState`
	`);`

Parameter	`EOperatingState in_OperatingState`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfPrimitiveDataType

Tabelle 7- 36 GetNameOfPrimitiveDataType() - Native C++

Syntax	`const WCHAR* GetNameOfPrimitiveDataType(`
	`  EPrimitiveDataType in_DataType`
	`);`

Parameter	`EPrimitiveDataType in_DataType`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfTagListDetails

Tabelle 7- 37 GetNameOfTagListDetails() - Native C++

Syntax	`const WCHAR* GetNameOfTagListDetails(`
	
	`  ETagListDetails in_TagListDetails`
	`);`

Parameter	`ETagListDetails in_TagListDetails`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfRuntimeConfigChanged()

Tabelle 7- 38 GetNameOfRuntimeConfigChanged() - Native C++

Syntax	`const WCHAR* GetNameOfRuntimeConfigChanged( ERuntimeConfigChanged in_RuntimeConfigChanged); );`
Parameter	`ERuntimeConfigChanged in_RuntimeConfigChanged:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfInstanceConfigChanged()

Tabelle 7- 39 GetNameOfInstanceConfigChanged() - Native C++

Syntax	`const WCHAR* GetNameOfInstanceConfigChanged( EInstanceConfigChanged in_InstanceConfigChanged); );`
Parameter	`EInstanceConfigChanged in_InstanceConfigChanged:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfDirection()

Tabelle 7- 40 GetNameOfDirection() - Native C++

Syntax	`const WCHAR* GetNameOfDirection( EDirection in_Direction );`
Parameter	`EDirection in_Direction:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfDiagSeverity()

Tabelle 7- 41 GetNameOfDiagSeverity() - Native C++

Syntax	`const WCHAR* GetNameOfDiagSeverity( EDiagSeverity in_DiagSeverity );`
Parameter	`EDiagSeverity in_DiagSeverity:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfRackOrStationFaultType()

Tabelle 7- 42 GetNameOfRackOrStationFaultType() - Native C++

Syntax	`const WCHAR* GetNameOfRackOrStationFaultType( ERackOrStationFaultType in_RackOrStationFaultType );`
Parameter	`ERackOrStationFaultType in_RackOrStationFaultType:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfProcessEventType()

Tabelle 7- 43 GetNameOfProcessEventType() - Native C++

Syntax	`const WCHAR*(GetNameOfProcessEventType( EProcessEventType in_ProcessEventType );`
Parameter	`EProcessEventType in_ProcessEventType`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfPullOrPlugEventType()

Tabelle 7- 44 GetNameOfPullOrPlugEventType() - Native C++

Syntax	`const WCHAR* GetNameOfPullOrPlugEventType( EPullOrPlugEventType in_PullOrPlugEventType );`
Parameter	`EPullOrPlugEventType in_PullOrPlugEventType`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfCycleTimeMonitoringMode()

Tabelle 7- 45 GetNameOfCycleTimeMonitoringMode() - Native C++

Syntax	`const WCHAR* GetNameOfCycleTimeMonitoringMode( ECycleTimeMonitoringMode in_CycleTimeMonitoringMode );`
Parameter	`ECycleTimeMonitoringMode in_CycleTimeMonitoringMode`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfDiagProperty()

Tabelle 7- 46 GetNameOfDiagProperty() - Native C++

Syntax	`const WCHAR* GetNameOfDiagProperty( EDiagProperty in_DiagProperty );`
Parameter	`EDiagProperty in_DiagProperty`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

GetNameOfAutodiscoverType()

Tabelle 7- 47 GetNameOfAutodiscoverType() - Native C++

Syntax	`const WCHAR* GetNameOfAutodiscoverType( EAutodiscoverType in_AutodiscoverType );`
Parameter	`EAutodiscoverType in_AutodiscoverType`: Aufzählungseintrag
Rückgabewerte	`const WCHAR*`: Name des Aufzählungseintrags

Siehe auch

EPrimitiveDataType (Seite 380)

EDataType (Seite 383)

Aufzählungen (Seite 369)

7.5 API ISimulationRuntimeManager

7.5.1 Schnittstellen - Informationen und Einstellungen

GetVersion() / Version { get; }

Liefert die Version des Runtime Managers zurück. Wenn die Funktion fehlschlägt, wird die Version 0.0 zurückgegeben.

Tabelle 7- 48 GetVersion() - Native C++

Syntax	`UINT32 GetVersion();`
Parameter	Keine
Rückgabewerte	`UINT32`: Runtime Manager Version (`HIWORD` = Major, `LOWORD` = Minor)

Tabelle 7- 49 Version { get; } - .NET (C#)

Syntax	`UInt32 Version { get; }`
Parameter	Keine
Rückgabewerte	`UInt32`: Runtime Manager Version (`HIWORD` = Major, `LOWORD` = Minor)

IsInitialized() / IsInitialized { get; }

Liefert einen Wert zurück, der anzeigt, ob die API erfolgreich initialisiert wurde.

Tabelle 7- 50 IsInitialized() - Native C++

Syntax	`bool IsInitialized();`
Parameter	Keine
Rückgabewerte	• `false`: Wenn die API nicht initialisiert wurde. • `true`: Wenn die API initialisiert wurde.

Tabelle 7- 51 IsInitialized { get; } - .NET (C#)

Syntax	`bool IsInitialized { get; }`
Parameter	Keine
Rückgabewerte	• `false`: Wenn die API nicht initialisiert wurde. • `true`: Wenn die API initialisiert wurde.

IsRuntimeManagerAvailable() / IsRuntimeManagerAvailable { get; }

Die Funktion gibt `false` zurück, wenn die Verbindung zum Runtime Manager unterbrochen ist. Dies passiert nur, wenn der Prozess des Runtime Managers geschlossen ist.

Abonnieren Sie das Ereignis `OnRuntimeManagerLost()`, um zu erfahren, ob die Verbindung unterbrochen ist. Siehe Ereignisse (Seite 144).

Tabelle 7- 52 IsRuntimeManagerAvailable() - Native C++

Syntax	`bool IsRuntimeManagerAvailable();`
Parameter	Keine
Rückgabewerte	• `false`: Wenn die Verbindung unterbrochen ist. • `true`: Wenn die Verbindung aktiv ist.

Tabelle 7- 53 IsRuntimeManagerAvailable { get; } - .NET (C#)

Syntax	`bool IsRuntimeManagerAvailable{ get; }`
Parameter	Keine
Rückgabewerte	• `false`: Wenn die Verbindung unterbrochen ist. • `true`: Wenn die Verbindung aktiv ist.

Shutdown()

Beendet die Kommunikation mit dem Runtime Manager und bereinigt die Schnittstellen.

Rufen Sie diese Funktion in folgenden Fällen auf:

- Unmittelbar bevor die API-Bibliothek (DLL) abgemeldet wird (Native C++).
- Wenn Ihre Applikation den Runtime Manager nicht mehr nutzt.

Tabelle 7- 54 Shutdown() - Native C++

Syntax	`ERuntimeErrorCode Shutdown()`	
Parameter	Keine	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.

Tabelle 7- 55 Shutdown() - .NET (C#)

Syntax	`void Shutdown()`
Parameter	Keine
Rückgabewerte	Keine

7.5.2 Simulation Runtime Instanzen

GetRegisteredInstancesCount()

Liefert die Anzahl der Instanzen zurück, die im Runtime Manager registriert sind. Wenn die Funktion fehlschlägt, ist der Rückgabewert 0.

Tabelle 7- 56 GetRegisteredInstancesCount() - Native C++

Syntax	`UINT32 GetRegisteredInstancesCount();`
Parameter	Keine
Rückgabewerte	`UINT32`: Anzahl der verfügbaren Instanzen.

GetRegisteredInstanceInfoAt()

Liefert die Information über eine bereits registrierte Instanz zurück. Sie können die ID oder den Namen verwenden, um eine Schnittstelle dieser Instanz zu erzeugen, siehe `CreateInterface()`.

Tabelle 7- 57 GetRegisteredInstanceInfoAt() - Native C++

Syntax	`ERuntimeErrorCode GetRegisteredInstanceInfoAt( UINT32 in_Index, SInstanceInfo* out_InstanceInfo );`	
Parameter	• `UINT32 in_Index:` Index der erzeugten Instanz, von der Sie die Information empfangen möchten. Der Index muss kleiner sein als der Wert, den Sie `GetRegisteredInstanceCount()` aufrufen. • `SInstanceInfo* out_InstanceInfo:` Die Information mit Name und ID der Instanz. Siehe Datentypen (Seite 344).	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_DOES_NOT_EXIST`	Es gibt keine Instanz-Information zu diesem Index.
	`SREC_INDEX OUT_OF_RANGE`	Der Index ist größer als 15.

RegisteredInstanceInfo { get; }

Liefert die Information über alle bereits registrierten Instanzen. Verwenden Sie die ID oder den Namen dieser Instanz, um eine Schnittstelle von dieser Instanz zu erzeugen, siehe `CreateInterface()`.

Tabelle 7- 58 RegisteredInstanceInfo { get; } - .NET (C#)

Syntax	`SInstanceInfo[] RegisteredInstanceInfo { get; }`	
Parameter	Keine	
Rückgabewerte	`SInstanceInfo[]`: Ein Array mit Informationen zu allen registrierten Instanzen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

RegisterInstance()

Registriert im Runtime Manager eine neue Instanz eines virtuellen Controllers. Erzeugt und liefert eine Schnittstelle von dieser Instanz zurück.

Tabelle 7- 59 RegisterInstance() - Native C++

Syntax	`ERuntimeErrorCode RegisterInstance(`
	`  IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode RegisterInstance(`
	`  WCHAR* in_InstanceName,`
	`  IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode RegisterInstance(`
	`  ECPUType in_CPUType,`
	`  IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode RegisterInstance(`
	`  ECPUType in_CPUType,`
	`  WCHAR* in_InstanceName,`
	`  IInstance** out_InstanceInterface`
	`);`	

Parameter	• `ECPUType in_CPUType:` Definiert, welcher CPU-Typ beim Start der Instanz simuliert wird. Die Voreinstellung ist `"SRCT_1500_Unspecified"`. Wenn über STEP 7 oder von der Virtual SIMATIC Memory Card ein anderer CPU-Typ geladen wird, dann gilt dieser CPU-Typ. • `WCHAR* in_InstanceName:` Name, den die Instanz erhalten soll. Jede Instanz muss einen eindeutigen Namen erhalten. Wenn kein Name vergeben wird beim Registrieren einer neuen Instanz, dann erhält die Instanz den Namen `"Instance_#"` (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name `"Instance_#.#"` verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als `DINSTANCE_NAME_LENGTH`. Siehe Datentypen (Seite 344). • `IInstance** out_InstanceInterface:` Zeiger auf einen Simulation Runtime Schnittstellenzeiger. Der Zeiger muss mit `NULL` initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Name oder der IInstance-Zeiger ist ungültig.
	`SREC_LIMIT_REACHED`	Es sind bereits 16 Instanzen im Runtime Manager registriert.
	`SREC_ALREADY_EXISTS`	Eine Instanz mit diesem Namen existiert bereits.

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`// Example: How To Create And Register An Instance`
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	`  result = api->RegisterInstance(&psa);`
	`}`

Hinweis

Native C++

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118).

Tabelle 7- 60 RegisterInstance() - .NET (C#)

Syntax	`IInstance RegisterInstance();`
	`IInstance RegisterInstance(`
	`  string in_InstanceName`
	`);`
	`IInstance RegisterInstance(`
	`  ECPUType in_CPUType`
	`);`
	`IInstance RegisterInstance(`
	`  ECPUType in_CPUType`
	`  string in_InstanceName`
	`);`	

Parameter	• `ECPUType in_CPUType:` Definiert, welcher CPU-Typ beim Start der Instanz simuliert wird. Die Voreinstellung ist `"ECPUType.Unspecified"`. Wenn über STEP 7 oder von der Virtual SIMATIC Memory Card ein anderer CPU-Typ geladen wird, dann gilt dieser CPU-Typ. • `string in_InstanceName:` Name, den die Instanz erhalten soll. Jede Instanz muss einen eindeutigen Namen erhalten. Wenn kein Name vergeben wird beim Registrieren einer neuen Instanz, dann erhält die Instanz den Namen `"Instance_#"` (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name `"Instance_#.#"` verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als `DINSTANCE_NAME_LENGTH`. Siehe Datentypen (Seite 344).	
Rückgabewerte	Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers, ansonsten einen Null-Zeiger.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Name ist ungültig.
	`ERuntimeErrorCode.LimitReached`	Es sind bereits 16 Instanzen im Runtime Manager registriert.
	`ERuntimeErrorCode.AlreadyExists`	Eine Instanz mit diesem Namen existiert bereits.

RegisterCustomInstance()

Registriert eine neue Instanz eines virtuellen Controllers im Runtime Manager. Erzeugt und liefert eine Schnittstelle dieser Instanz zurück.

Tabelle 7- 61 RegisterCustomInstance() - Native C++

Syntax	`ERuntimeErrorCode RegisterCustomInstance(`
	` WCHAR* in_VplcDll,`
	` IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode RegisterCustomInstance(`
	` WCHAR* in_VplcDll,`
	` WCHAR* in_InstanceName,`
	` IInstance** out_InstanceInterface`
	`);`	

Parameter	• `WCHAR* in_VplcDll:` Der vollständige Pfad zur DLL des virtuellen Controllers, den die "Siemens.Simatic.Simulation.Runtime.Instance.exe" bei PowerOn laden wird. • `WCHAR* in_InstanceName:` Name, den die Instanz erhalten soll. Jede Instanz muss einen eindeutigen Namen erhalten. Wenn kein Name vergeben wird beim Registrieren einer neuen Instanz, dann erhält die Instanz den Namen `"Instance_#"` (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name `"Instance_#.#"` verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als `DINSTANCE_NAME_LENGTH`. Siehe Datentypen (Seite 344). • `IInstance** out_InstanceInterface:` Zeiger auf einen Simulation Runtime Schnittstellenzeiger. Der Zeiger muss mit `NULL` initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der DLL-Name, der Instanzname oder der IInstance-Zeiger ist ungültig.
	`SREC_LIMIT_REACHED`	Es sind bereits 16 Instanzen im Runtime Manager registriert.
	`SREC_ALREADY_EXISTS`	Eine Instanz mit diesem Namen existiert bereits.

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`// Example: How To Create And Register An Instance`
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	` result = api->`**RegisterCustomInstance**`(L"C:\\Temp\\vplc.dll");`
	`}`

Hinweis

Native C++

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118).

Tabelle 7- 62 RegisterCustomInstance() - .NET (C#)

Syntax	`IInstance RegisterCustomInstance(`
	` string in_VplcDll`
	`);`
	`IInstance RegisterCustomInstance(`
	` string in_VplcDll,`
	` string in_InstanceName`
	`);`	

Parameter	• `string in_VplcDll:` Der vollständige Pfad zur DLL des virtuellen Controllers, den die "Siemens.Simatic.Simulation.Runtime.Instance.exe" bei PowerOn laden wird. • `string in_InstanceName:` Name, den die Instanz erhalten soll. Jede Instanz muss einen eindeutigen Namen erhalten. Wenn kein Name vergeben wird beim Registrieren einer neuen Instanz, dann erhält die Instanz den Namen `"Instance_#"` (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name `"Instance_#.#"` verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als `DINSTANCE_NAME_LENGTH`. Siehe Datentypen (Seite 344).	
Rückgabewerte	Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers, ansonsten ein Null-Zeiger.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Name oder die ID ist ungültig.
	`ERuntimeErrorCode.LimitReached`	Es sind bereits 16 Instanzen im Runtime Manager registriert.
	`ERuntimeErrorCode.AlreadyExists`	Eine Instanz mit diesem Namen existiert bereits.

CreateInterface()

Erzeugt und liefert eine Schnittstelle einer bereits registrierten Instanz eines virtuellen Controllers zurück.

Die Instanz kann über die Anwendung registriert worden sein oder über eine andere Anwendung, die die Simulation Runtime API nutzt.

Tabelle 7- 63 CreateInterface() - Native C++

Syntax	`ERuntimeErrorCode CreateInterface(`
	` WCHAR* in_InstanceName,`
	` IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode CreateInterface(`
	` INT32 in_InstanceID,`
	` IInstance** out_InstanceInterface`
	`);`	

Parameter	• `INT32 in_InstanceID:` Die ID der registrierten Instanz, von der Sie die Schnittstelle empfangen möchten. • `WCHAR* in_InstanceName:` Der Name der registrierten Instanz, von der Sie die Schnittstelle empfangen möchten. • `IInstance** out_InstanceInterface:` Zeiger auf einen Simulation Runtime Schnittstellenzeiger. Der Zeiger **muss** mit `NULL` initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Name, die ID oder der IInstance-Zeiger ist ungültig.
	`SREC_DOES_NOT_EXIST`	Die Instanz ist nicht im Runtime Manager registriert.

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`IInstance* psa1 = NULL;`
	`IInstance* psa2 = NULL;`
	`if (result == SREC_OK)`
	`{`
	` result = api->CreateInterface(0, &psa1);`
	` result = api->CreateInterface(0, &psa2); // psa2 will be the same as psa1`
	`}`	

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	` result = api->CreateInterface(L"My SimulationRuntime Instance", &psa);`
	`}`	

Hinweis

Native C++

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118).

Tabelle 7- 64 CreateInterface() - .NET (C#)

Syntax	`IInstance CreateInterface(`
	` string in_InstanceName`
	`);`
	`IInstance CreateInterface(`
	` INT32 in_InstanceID`
	`);`	

Parameter	• `INT32 in_InstanceID:` Die ID der registrierten Instanz, von der Sie die Schnittstelle empfangen möchten. • `string in_InstanceName:` Der Name der registrierten Instanz, von der Sie die Schnittstelle empfangen möchten.	
Rückgabewerte	Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers, ansonsten ein Null-Zeiger.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Name oder die ID ist ungültig.
	`ERuntimeErrorCode.DoesNotExists`	Die Instanz ist nicht im Runtime Manager registriert.

7.5.3 Remote-Verbindungen

OpenPort()

Öffnet einen Port, mit dem sich ein weiterer Runtime Manager verbinden kann.

Tabelle 7- 65 OpenPort() - Native C++

Syntax	`ERuntimeErrorCode OpenPort( UINT16 in_Port );`	
Parameter	• `UINT16 in_Port:` Der Port. Der Wert muss größer sein als 1024.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_ALREADY_EXISTS`	Es ist bereits ein Port offen.
	`SREC_WRONG_ARGUMENT`	Der Port ist ungültig.
	`SREC_CONNECTION_ERROR`	Der Port kann nicht geöffnet werden.

Tabelle 7- 66 OpenPort() - .NET (C#)

Syntax	`void OpenPort( UInt16 in_Port );`	
Parameter	• `UInt16 in_Port:` Der Port. Der Wert muss größer sein als 1024.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.AlreadyExists`	Es ist bereits ein Port offen.
	`ERuntimeErrorCode.WrongArgument`	Der Port ist ungültig.
	`ERuntimeErrorCode.ConnectionError`	Der Port kann nicht geöffnet werden.

ClosePort()

Schließt einen offenen Port und alle offenen Verbindungen, die ein weiterer Runtime Manager zu diesem offenen Port erstellt hat.

Tabelle 7- 67 ClosePort() - Native C++

Syntax	`ERuntimeErrorCode ClosePort();`	
Parameter	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WARNING_INVALID_CALL`	Es ist kein Port offen.

Tabelle 7- 68 ClosePort() - .NET (C#)

Syntax	`void ClosePort(`
	`  UInt16 in_Port`
	`);`	

Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

GetPort() / Port { get; }

Liefert den offenen Port zurück. Wenn kein Port offen ist oder die Funktion fehlschlägt, dann ist der Rückgabewert 0.

Tabelle 7- 69 GetPort() - Native C++

Syntax	`UINT16 GetPort();`
Parameter	Keine
Rückgabewerte	`UINT16`: Der offene Port. 0, wenn kein Port offen ist.

Tabelle 7- 70 Port { get; } - .NET (C#)

Syntax	`UInt16 Port { get; }`
Parameter	Keine
Rückgabewerte	`UInt16`: Der offene Port. 0, wenn kein Port offen ist.
Ausnahmen	Keine

GetRemoteConnectionsCount()

Liefert die Anzahl der offenen Remote-Verbindungen.

Tabelle 7- 71 GetRemoteConnectionsCount() - Native C++

Syntax	`UINT32 GetRemoteConnectionsCount();`
Parameter	Keine
Rückgabewerte	`UINT32`: Anzahl der offenen Remote-Verbindungen.

GetRemoteConnectionInfoAt()

Liefert die Information zu einer offenen Verbindung.

Tabelle 7- 72 GetRemoteConnectionInfoAt()- Native C++

Syntax	`ERuntimeErrorCode GetRemoteConnectionInfoAt(`
	` UINT32 in_Index,`
	` SConnectionInfo* out_ConnectionInfo`
	`);`	

Parameter	• `UINT32 in_Index:` Index der Verbindungs-Information, die erwartet wird. • `SConnectionInfo* out_ConnectionInfo:` Die Verbindungs-Information für diesen Index.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INDEX_OUT_OF_RANGE`	Eine Verbindungs-Information zu diesem Index existiert nicht.

RemoteConnectionInfo { get; }

Liefert ein Array von Informationen zu allen offenen Verbindungen zurück.

Tabelle 7- 73 RemoteConnectionInfo { get; } - .NET (C#)

Syntax	`SConnectionInfo[] RemoteConnectionInfo { get; }`	
Parameter	Keine	
Rückgabewerte	`SConnectionInfo[]`: Ein Array von Informationen zu allen offenen Verbindungen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

RemoteConnect()

Erstellt eine neue Verbindung zu einem Remote Runtime Manager oder nutzt eine bereits existierende Verbindung, um eine IRemoteRuntimeManager Schnittstelle zu erstellen.

Tabelle 7- 74 RemoteConnect() - Native C++

Syntax	`ERuntimeErrorCode RemoteConnect(`
	` UINT8 in_IP3,`
	` UINT8 in_IP2,`
	` UINT8 in_IP1,`
	` UINT8 in_IP0,`
	` UINT16 in_Port,`
	` IRemoteRuntimeManager** out_RemoteRuntimeManagerInterface`
	`ERuntimeErrorCode RemoteConnect(`
	` UIP in_IP,`
	` UINT16 in_Port,`
	` IRemoteRuntimeManager** out_RunTimeManagerInterface`
	`);`	

Parameter	• `UINT8 in_IP3:` Erster Teil der IP-Adresse des Remote-PC. • `UINT8 in_IP2:` Zweiter Teil der IP-Adresse des Remote-PC. • `UINT8 in_IP1:` Dritter Teil der IP-Adresse des Remote-PC. `UINT8 in_IP0:` Letzter Teil der IP-Adresse des Remote-PC. • `UIP in_IP:` IP-Adresse des Remote-PC. • `UINT16 in_Port:` Der Port, der auf dem Remote-PC geöffnet ist. • `IRemoteRuntimeManager** out_RemoteRuntimeManagerInterface:` Zeiger auf einen Remote Runtime Manager Schnittstellenzeiger. Der Zeiger **muss** mit `NULL` initialisiert werden. Die Schnittstelle wird in der Funktion erzeugt.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_CONNECTION_ERROR`	Die Verbindung zum Remote Runtime Manager kann nicht hergestellt werden.
	`SREC_WRONG_ARGUMENT`	IP, Port oder IInstance-Zeiger ist ungültig.
	`SREC_WRONG_VERSION`	Die Version der API ist nicht kompatibel mit der Runtime. Siehe Kompatibilität beim Upgrade (Seite 33).

Beispiel C++	`ISimulationRuntimeManager* api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`IRemoteRuntimeManager * client = NULL;`
	
	`if (result == SREC_OK)`
	`{`
	`result = api->RemoteConnect(192,203,145,144, 4444, &client);`
	`}`

Hinweis

Native C++

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118).

Tabelle 7- 75 RemoteConnect() - .NET (C#)

Syntax	`IRemoteRuntimeManager RemoteConnect(`
	`string in_ConnectionString`
	`);`
	`IRemoteRuntimeManager RemoteConnect(`
	`SIP in_IP,`
	`UInt16 in_Port`
	`);`
	`IRemoteRuntimeManager RemoteConnect(`
	`Byte in_IP3,`
	`Byte in_IP2,`
	`Byte in_IP1,`
	`Byte in_IP0,`
	`UInt16 in_Port`
	`);`	

Parameter	• `Byte in_IP3:` Erster Teil der IP-Adresse des Remote-PC. • `Byte in_IP2:` Zweiter Teil der IP-Adresse des Remote-PC. • `Byte in_IP1:` Dritter Teil der IP-Adresse des Remote-PC. • `Byte in_IP0:` Letzter Teil der IP-Adresse des Remote-PC. • `string in_ConnectionString:` Ein String in Form von "<IP3>.<IP2>.<IP1>.<IP0>:<Port>" Beispiel: "182.203.145.144:4444". • `SIP in_IP:` IP-Adresse des Remote-PC. • `UInt16 in_Port:` Der Port, der auf dem Remote-PC geöffnet ist.	
Rückgabewerte	`IRemoteRuntimeManager:` Schnittstelle zum Remote Runtime Manager.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung

	`ERuntimeErrorCode.ConnectionError`	Verbindung zum Remote Runtime Manager kann nicht hergestellt werden.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	IP oder Port ist ungültig.
	`ERuntimeErrorCode.WrongVersion`	Die Version der API ist nicht kompatibel mit der Runtime. Siehe Kompatibilität beim Upgrade (Seite 33).

Siehe auch

Variablentabelle (Seite 179)

Datentypen (Seite 316)

7.5.3.1 RunAutodiscover()

Beschreibung

Mit dieser Funktion werden alle Runtime Manager identifiziert, die sich im Netzwerk befinden und die bereit sind, eine Remote-Verbindung herzustellen.

Hinweis

Die Funktion identifiziert Runtime Manager ab PLCSIM Advanced V3.0.

Voraussetzungen

- Der Runtime Manager muss laufen und Remote-Verbindungen zulassen.
- Die Firewall des Remote-PC darf den Verkehr am ausgewählten UDP-Port nicht blockieren.
- Geräte im lokalen Netzwerk (z. B. Router, Switches, Firewalls) dürfen Multicast Pakete der ausgewählten Klasse nicht blockieren.

RunAutodiscover()

Die Funktion startet im Netzwerk die Identifizierung der Runtime Manager.

Tabelle 7- 76 RunAutodiscover() - Native C++

Syntax	`ERuntimeErrorCode RunAutodiscover( UINT32 in_Timeout = 2000 );`	
Parameter	• `UINT32 in_Timeout` Ein Timeout-Wert in Millisekunden, der definiert, wie lange der lokale Runtime Manager auf Antworten vom Remote Manager wartet. Gültig ist ein Wert zwischen 500 ms und 30000 ms. Voreinstellung: 2000 ms.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WRONG_ARGUMENT`	Der Timeout-Wert ist außerhalb des zulässigen Bereichs.
	`SREC_AUTODISCOVER_ALREADY_RUNNING`	Ein Aufruf `RunAutodiscover()` läuft bereits im Hintergrund. Warten Sie auf die Meldung `SRRSI_DISCOVER_FINISHED` in der Callback-Funktion. Siehe EAutodiscoverType (Seite 393).
	`SREC_TIMEOUT`	Kommunikationsfehler im lokalen Runtime Manager.

Tabelle 7- 77 RunAutodiscover() - .NET (C#)

Syntax	`void RunAutodiscover( UInt32 in_Timeout = 2000 );`	
Parameter	• `UInt32 in_Timeout` Ein Timeout-Wert in Millisekunden, der definiert, wie lange der lokale Runtime Manager auf Antworten vom Remote Manager wartet. Gültig ist ein Wert zwischen 500 ms und 30000 ms. Voreinstellung: 2000 ms.	
Rückgabewerte	keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.WrongArgument`	Der Timeout-Wert ist außerhalb des zulässigen Bereichs.
	`ERuntimeErrorCode.AutodiscoverAlreadyRunning`	Ein Aufruf `RunAutodiscover()` läuft bereits im Hintergrund. Warten Sie auf die Meldung `AutodiscoverFinished` in der Callback-Funktion. Siehe EAutodiscoverType (Seite 393).
	`ERuntimeErrorCode.Timeout`	Kommunikationsfehler im lokalen Runtime Manager.

7.5.4 Ereignisse für ISimulationRuntimeManager

Ereignisse für Runtime Instanzen und Remote-Verbindungen

Für die Schnittstelle ISimulationRuntimeManager werden folgende Ereignisse ausgelöst:

Tabelle 7- 78 Ereignisse für ISimulationRuntimeManager

Ereignis	Ursache
OnConfigurationChanged (Seite 144)	Die Runtime Manager Konfiguration hat sich geändert: • Eine neue Instanz wird registriert. • Eine Instanz wird entfernt. • Eine Verbindung zu einem Client wird aufgebaut. Das Control Panel nutzt ein solches Ereignis, um die Liste der verfügbaren Instanzen zu aktualisieren.
OnRuntimeManagerLost (Seite 147)	Die Verbindung zum Runtime Manager ist unterbrochen.
RunAutodiscover (Seite 150)	Im Netzwerk wird nach Runtime Managern gesucht, die bereit sind, eine Remote-Verbindung herzustellen.

7.5.4.1 Ereignisse OnConfigurationChanged

OnConfigurationChanged

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 79 OnConfigurationChanged - .NET (C#)

Syntax	`event Delegate_SRCC_UINT32_UINT32_INT32 OnConfigurationChanged;`
Parameter	Keine. Siehe Delegate_SRCC_UINT32_UINT32_INT32 (Seite 332).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnConfigurationChangedCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Die Registrierung einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 80 RegisterOnConfigurationChangedCallback() - Native C++

Syntax	`void RegisterOnConfigurationChangedCallback( EventCallback_SRCC_UINT32_UINT32_INT32 in_CallbackFunction );`
Parameter	• `EventCallback_SRCC_UINT32_UINT32_INT32 in_CallbackFunction`: Eine Callback-Funktion, um ein Ereignis zu abonnieren. Siehe EventCallback_SRCC_UINT32_UINT32_INT32 (Seite 319).
Rückgabewerte	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnConfigurationChangedEvent()

Wenn das Ereignis eintritt, wird das registrierte Event-Objekt in den signalisierten Zustand gesetzt. Es kann nur ein Event-Objekt für das Ereignis registriert sein. Die Registrierung eines neuen Event-Objekts führt zum Löschen des vorhergehenden.

Tabelle 7- 81 RegisterOnConfigurationChangedEvent() - Native C++

Syntax	`void RegisterOnConfigurationChangedEvent();` `void RegisterOnConfigurationChangedEvent( HANDLE* in_Event );`
Parameter	• `None`: Ein internes Event-Objekt wird registriert. • `HANDLE* in_Event`: Ein Handle zu einem anwenderspezifischen Event-Objekt. Das Event-Objekt wird registriert.
Rückgabewerte	Keine

Tabelle 7- 82 RegisterOnConfigurationChangedEvent() - .NET (C#)

Syntax	`void RegisterOnConfigurationChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnConfigurationChangedCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 83 UnregisterOnConfigurationChangedCallback() - Native C++

Syntax	`void UnregisterOnConfigurationChangedCallback();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnConfigurationChangedEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 84 UnregisterOnConfigurationChangedEvent() - Native C++

Syntax	`void UnregisterOnConfigurationChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

Tabelle 7- 85 UnregisterOnConfigurationChangedEvent() - .NET (C#)

Syntax	`void UnregisterOnConfigurationChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

WaitForOnConfigurationChangedEvent()

Die Funktion blockiert das Programm solange, bis das registrierte Event-Objekt in den signalisierten Zustand gesetzt ist oder bis das Timeout-Intervall überschritten wird.

Tabelle 7- 86 WaitForOnConfigurationChangedEvent() - Native C++

Syntax	`bool WaitForOnConfigurationChangedEvent();`
	`bool WaitForOnConfigurationChangedEvent(`
	`  UINT32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UINT32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Tabelle 7- 87 WaitForOnConfigurationChangedEvent - .NET (C#)

Syntax	`bool WaitForOnConfigurationChangedEvent();`
	`bool WaitForOnConfigurationChangedEvent(`
	`  UInt32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UInt32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

7.5.4.2 Ereignisse OnRuntimeManagerLost

OnRuntimeManagerLost

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 88 OnRuntimeManagerLost - .NET (C#)

Syntax	`event Delegate_Void OnRuntimeManagerLost;`
Parameter	Keine. Siehe Delegate_Void (Seite 331).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnRuntimeManagerLostCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Die Registrierung einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 89 RegisterOnRuntimeManagerLostCallback() - Native C++

Syntax	`void RegisterOnRuntimeManagerLostCallback(`
	`  EventCallback_VOID in_CallbackFunction`
	`);`

Parameter	• `EventCallback_VOID in_CallbackFunction:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_VOID (Seite 318).
Rückgabewerte	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnRuntimeManagerLostEvent()

Wenn das Ereignis eintritt, wird das registrierte Event-Objekt in den signalisierten Zustand gesetzt. Es kann nur ein Event-Objekt für das Ereignis registriert sein. Die Registrierung eines neuen Event-Objekts führt zum Löschen des vorhergehenden.

Tabelle 7- 90 RegisterOnRuntimeManagerLostEvent() - Native C++

Syntax	`void RegisterOnRuntimeManagerLostEvent();`
	`void RegisterOnRuntimeManagerLostEvent(`
	`HANDLE* in_Event`
	`);`

Parameter	• `None:` Ein interner Event-Handle wird registriert. • `HANDLE* in_Event:` Ein anwenderspezifischer Event-Handle wird registriert.
Rückgabewerte	Keine

Tabelle 7- 91 RegisterOnRuntimeManagerLostEvent() - .NET (C#)

Syntax	`void RegisterOnRuntimeManagerLostEvent();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnRuntimeManagerLostCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 92 UnregisterOnRuntimeManagerLostCallback() - Native C++

Syntax	`void UnregisterOnRuntimeManagerLostCallback();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnRuntimeManagerLostEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 93 UnregisterOnRuntimeManagerLostEvent() - Native C++

Syntax	`void UnregisterOnRuntimeManagerLostEvent();`
Parameter	Keine
Rückgabewerte	Keine

Tabelle 7- 94 UnregisterOnRuntimeManagerLostEvent() - .NET (C#)

Syntax	`void UnregisterOnRuntimeManagerLostEvent();`
Parameter	Keine
Rückgabewerte	Keine

WaitForOnRuntimeManagerLostEvent()

Die Funktion wird das Programm solange blockieren, bis das registrierte Event-Objekt in den signalisierten Zustand gesetzt ist oder bis das Timeout-Intervall überschritten wird.

Tabelle 7- 95 WaitForOnRuntimeManagerLostEvent() - Native C++

Syntax	`bool WaitForOnRuntimeManagerLostEvent();`
	`bool WaitForOnRuntimeManagerLostEvent(`
	`  UINT32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UINT32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Timeout-Intervalls kein Ereignis empfangen wurde.

Tabelle 7- 96 WaitForOnRuntimeManagerLostEvent() - .NET (C#)

Syntax	`bool WaitForOnRuntimeManagerLostEvent();`
	`bool WaitForOnRuntimeManagerLostEvent(`
	`  UInt32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UInt32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true`: Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false`: Wenn während des definierten Timeout-Intervalls kein Ereignis empfangen wurde.

7.5.4.3 Ereignisse OnAutodiscoverData

OnAutodiscoverData

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 97 OnAutodiscoverData - .NET (C#)

Syntax	`event Delegate_SRRSI_AD OnAutodiscoverData`
Parameter	Keine. Siehe Delegate_SRRSI_AD (Seite 333).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnAutodiscoverCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Die Registrierung einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 98 RegisterOnAutodiscoverCallback() - Native C++

Syntax	`void RegisterOnAutodiscoverCallback(`
	`  EventCallback_SRRSI_AD in_CallbackFunction`
	`);`

Parameter	• `EventCallback_SRRSI_AD in_CallbackFunction:` Zeiger auf eine anwenderdefinierte Callback-Funktion. Siehe EventCallback_SRRSI_AD (Seite 320).
Rückgabewerte	Keine

UnregisterOnAutodiscoverCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 99 UnregisterOnAutodiscoverCallback() - Native C++

Syntax	`void UnregisterOnAutodiscoverCallback( );`
Parameter	Keine
Rückgabewerte	Keine

7.6 API IInstances

7.6.1 Schnittstellen - Informationen und Einstellungen

Dispose()

Löscht die managed Schnittstelle und entlädt die nativen Komponenten der Anwenderschnittstellen.

Tabelle 7- 100 Dispose() - .NET (C#)

Syntax	`void Dispose()`
Parameter	Keine
Rückgabewerte	Keine

GetID() / ID { get; }

Liefert die Instanz-ID zurück. Die ID wird vom Runtime Manager zugewiesen, wenn die Instanz registriert wird.

Tabelle 7- 101 GetID() - Native C++

Syntax	`INT32 GetID();`
Parameter	Keine
Rückgabewerte	`INT32`: Instanz-ID

Tabelle 7- 102 ID { get; } - .NET (C#)

Syntax	`UInt32 ID { get; }`
Parameter	Keine
Rückgabewerte	`UInt32`: Instanz-ID
Ausnahmen	Keine

GetName() / Name { get; }

Liefert den Namen der Instanz zurück.

Tabelle 7- 103 GetName() - Native C++

Syntax	`ERuntimeErrorCode GetName(`
	` WCHAR inout_Name[],`
	` UINT32 in_ArrayLength`
	`);`	

Parameter	• `WCHAR inout_Name[]:` Ein benutzerallokierter Speicher für den Namen der Instanz. Die Feldlänge soll mindestens so groß sein wie `DINSTANCE_NAME_MAX_LENGTH`. Siehe Definitionen und Konstanten (Seite 344). • `UINT32 in_ArrayLength:` Feldlänge (Wide-Character)	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_WRONG_ARGUMENT`	Der Name passt nicht in den Speicher.

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	` result = api->RegisterInstance(&psa);`
	`}`
	
	`WCHAR name[DINSTANCE_NAME_MAX_LENGTH];`
	`if (result == SREC_OK)`
	`{`
	` result = psa->GetName(name, DINSTANCE_NAME_MAX_LENGTH);`
	`}`	

Tabelle 7- 104 Name { get; } - .NET (C#)

Syntax	`string Name { get; }`	
Parameter	Keine	
Rückgabewerte	Name der Instanz.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.

GetCPUType()

Liefert den CPU-Typ des virtuellen Controllers zurück.

Tabelle 7- 105 GetCPUType() - Native C++

Syntax	`ECPUType GetCPUType();`
Parameter	Keine
Rückgabewerte	Ein Aufzählungselement, das den CPU-Typ definiert. Siehe ECPUType (Seite 375).

SetCPUType()

Setzt den CPU-Typ des virtuellen Controllers. Ein Wechsel des CPU-Typs erfolgt nur bei einem Neustart des Controllers.

Tabelle 7- 106 SetCPUType() - Native C++

Syntax	`void SetCPUType(ECPUType in_Value);`
Parameter	• `ECPUType in_Value:` Definiert, welcher CPU-Typ beim Start der Instanz simuliert wird. Wenn über STEP 7 oder von der Virtual Memory Card ein anderer CPU-Typ geladen wird, dann gilt dieser CPU-Typ.
Rückgabewerte	Keine

CPUType { get; set; }

Liefert oder setzt den CPU-Typ des virtuellen Controllers. Ein Wechsel des CPU-Typs erfolgt nur bei einem Neustart des Controllers.

Wenn über STEP 7 oder von der Virtual Memory Card ein anderer CPU-Typ geladen wird, dann gilt dieser CPU-Typ.

Tabelle 7- 107 CPUType { get; set; } - .NET (C#)

Syntax	`ECPUType CPUType { get; set; }`
Parameter	Keine
Rückgabewerte	Ein Aufzählungselement, das den CPU-Typ definiert.
Ausnahmen	Keine

GetCommunicationInterface()

Liefert die Kommunikationsschnittstelle des virtuellen Controllers zurück: Lokale Kommunikation (Softbus) oder TCPIP. Ein Wechsel der Kommunikationsschnittstelle erfolgt nur bei einem Neustart des Controllers. Alle Instanzen, die gestartet werden, müssen die selbe Kommunikationsschnittstelle nutzen.

PowerOn wird verhindert, wenn eine Kommunikationsschnittstelle ausgewählt ist, die nicht von den gestarteten Instanzen genutzt wird.

Tabelle 7- 108 GetCommunicationInterface() - Native C++

Syntax	`ECommunicationInterface GetCommunicationInterface();`
Parameter	Keine
Rückgabewerte	• `SRCI_NONE` Kann nicht ausgewählt werden. Wird zurückgegeben, wenn die Instanz-Schnittstelle nicht mehr gültig ist. • `SRCI_SOFTBUS` Wird zurückgegeben, wenn der virtuelle Controller den Softbus nutzt. • `SRCI_TCPIP` Wird zurückgegeben, wenn der virtuelle Controller über den virtuellen Adapter kommuniziert.

SetCommunicationInterface()

Setzt die Kommunikationsschnittstelle des virtuellen Controllers: Lokale Kommunikation (Softbus) oder TCPIP. Ein Wechsel der Kommunikationsschnittstelle erfolgt nur bei einem Neustart des Controllers. Alle Instanzen, die gestartet werden, müssen die selbe Kommunikationsschnittstelle nutzen.

PowerOn wird verhindert, wenn eine Kommunikationsschnittstelle ausgewählt ist, die nicht von den gestarteten Instanzen genutzt wird.

Tabelle 7- 109 SetCommunicationInterface() - Native C++

Syntax	`void SetCommunicationInterface(ECommunicationInterface in_Value);`
Parameter	• `SRCI_NONE` Kann nicht ausgewählt werden. • `SRCI_SOFTBUS` Wird gesetzt, um die Kommunikation über den Softbus zu aktivieren. • `SRCI_TCPIP` Wird gesetzt, um die Kommunikation über den virtuellen Adapter zu aktivieren.
Rückgabewerte	Keine

CommunicationInterface { get; set; }

Setzt die Kommunikationsschnittstelle des virtuellen Controllers oder liefert sie zurück: Lokale Kommunikation (Softbus) oder TCPIP. Ein Wechsel der Kommunikationsschnittstelle erfolgt nur beim Neustart des Controllers. Alle Instanzen, die gestartet werden, müssen die selbe Kommunikationsschnittstelle nutzen.

PowerOn wird verhindert, wenn eine Kommunikationsschnittstelle ausgewählt ist, die nicht von den gestarteten Instanzen genutzt wird.

Tabelle 7- 110 CommunicationInterface { get; set; } - .NET (C#)

Syntax	`ECommunicationInterface CommunicationInterface { get; set; }`
Parameter	Keine
Rückgabewerte	• `ECommunicationInterface.None` Kann nicht ausgewählt werden. Wird zurückgegeben, wenn die Instanz-Schnittstelle nicht mehr gültig ist. • `ECommunicationInterface.Softbus` Wird zurückgegeben, wenn der virtuelle Controller den Softbus nutzt. • `ECommunicationInterface.TCPIP` Wird zurückgegeben, wenn der virtuelle Controller über den virtuellen Adapter kommuniziert.
Ausnahmen	Keine

GetInfo() / Info { get; }

Liefert eine Struktur, die Informationen über die Instanz bereitstellt.

Tabelle 7- 111 GetInfo() - Native C++

Syntax	`SInstanceInfo GetInfo();`
Parameter	Keine
Rückgabewerte	`SInstanceInfo`: Eine Struktur, die Informationen über die Instanz bereitstellt. Siehe SInstanceInfo (Seite 354).

Tabelle 7- 112 Info { get; } - .NET (C#)

Syntax	`SInstanceInfo Info { get; }`
Parameter	Keine
Rückgabewerte	`SInstanceInfo`: Eine Struktur, die Informationen über die Instanz bereitstellt.
Ausnahmen	Keine

UnregisterInstance()

Meldet diese Instanz vom Runtime Manager ab.

Hinweis

Verlust der Schnittstellen

Andere Anwendungen, die mit dieser Instanz verbunden sind, werden ihre Schnittstelle zu dieser Instanz verlieren.

Tabelle 7- 113 UnregisterInstance() - Native C++

Syntax	`ERuntimeErrorCode UnregisterInstance();`	
Parameter	Keine	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 114 UnregisterInstance() - .NET (C#)

Syntax	`void UnregisterInstance();`	
Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.2 Controller - Informationen und Einstellungen

GetControllerName() / ControllerName { get; }

Liefert den heruntergeladenen Namen des virtuellen Controllers zurück.

Tabelle 7- 115 GetControllerName() - Native C++

Syntax	`ERuntimeErrorCode GetControllerName(`
	`  WCHAR inout_Name[],`
	`  UINT32 in_ArrayLength`
	`);`	

Parameter	• `WCHAR inout_Name[]:` Ein benutzerallokierter Speicher für den Namen. • `UINT32 in_ArrayLength:` Die Länge des Speichers.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INDEX_OUT_OF_RANGE`	Der Name passt nicht in den Speicher.

Tabelle 7- 116 ControllerName { get; } - .NET (C#)

Syntax	`string ControllerName { get; }`	
Parameter	Keine	
Rückgabewerte	`string:` Der heruntergeladene Name des virtuellen Controllers.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

GetControllerShortDesignation() / ControllerShortDesignation { get; }

Liefert die heruntergeladene Kurzbezeichnung des virtuellen Controllers zurück.

Tabelle 7- 117 GetControllerShortDesignation() - Native C++

Syntax	`ERuntimeErrorCode GetControllerShortDesignation(`
	`  WCHAR inout_ShortDesignation[],`
	`  UINT32 in_ArrayLength`
	`);`	

Parameter	• `WCHAR inout_ShortDesignation[]:` Ein benutzerallokierter Speicher für die Kurzbezeichnung. • `UINT32 in_ArrayLength:` Die Länge des Speichers.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INDEX_OUT_OF_RANGE`	Der Name passt nicht in den Speicher.

Tabelle 7- 118 ControllerShortDesignation { get; } - .NET (C#)

Syntax	`string ControllerShortDesignation { get; }`	
Parameter	Keine	
Rückgabewerte	`string:` Die heruntergeladene Kurzbezeichnung des virtuellen Controllers.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

GetControllerIPCount()

Liefert die Anzahl der konfigurierten IP-Adressen des virtuellen Controllers zurück. Wenn die Funktion fehlschlägt, ist der Rückgabewert 0.

Tabelle 7- 119 GetControllerIPCount() - Native C++

Syntax	`UINT32 GetControllerIPCount();`
Parameter	Keine
Rückgabewerte	`INT32:` Anzahl der konfigurierten IP-Adressen des virtuellen Controllers.

GetControllerIP() / ControllerIP { get; }

Liefert eine konfigurierte IP-Adresse der Instanz zurück.

Tabelle 7- 120 GetControllerIP() - Native C++

Syntax	`UIP GetControllerIP();`
	`UIP GetControllerIP(`
	`  UINT32 in_Index`
	`);`

Parameter	• `WCHAR in_Index:` Der Index der IP-Adresse, die Sie erhalten möchten. Der Index muss kleiner sein als der Wert, den Sie von `GetControllerIPCount()` erhalten. Die Voreinstellung ist 0.
Rückgabewerte	`UIP`: IP-Adresse des virtuellen Controllers. Wenn die Funktion fehlschlägt, ist der Rückgabewert 0.

Tabelle 7- 121 ControllerIP { get; } - .NET (C#)

Syntax	`string[] ControllerIP { get; }`
Parameter	Keine
Rückgabewerte	`string`: Alle heruntergeladenen IP-Adressen des virtuellen Controllers. Wenn die Funktion fehlschlägt, ist das Feld leer.
Ausnahmen	Keine

GetControllerIPSuite4() / ControllerIPSuite4 { get; }

Liefert die IP-Suite der Instanz zurück. Wenn die Kommunikationsschnittstelle "Softbus" genutzt wird, sind Subnetzmaske und Standard-Gateway 0.

Tabelle 7- 122 GetControllerIPSuite4() Native C++

Syntax	`SIPSuite4 GetControllerIPSuite4();`
	`SIPSuite4 GetControllerIPSuite4(`
	`  UINT32 in_Index`
	`);`

Parameter	• `WCHAR in_Index:` Der Index der IP-Adresse, die Sie erhalten möchten. Der Index muss kleiner sein als der Wert, den Sie von `GetControllerIPCount()` erhalten. Die Voreinstellung ist 0.
Rückgabewerte	`SIPSuite4`: Die IP-Suite des virtuellen Controllers. Wenn die Funktion fehlschlägt, sind die Rückgabewerte 0.

Tabelle 7- 123 ControllerIPSuite4 { get; } - .NET (#)

Syntax	`SIPSuite4[] ControllerIPSuite4 { get; };`
Parameter	Keine
Rückgabewerte	`SIPSuite4[]`: Alle heruntergeladenen IP-Suites des virtuellen Controllers. Wenn die Funktion fehlschlägt, ist das Feld leer.
Ausnahmen	Keine

SetIPSuite()

Setzt die IP-Suite der Netzwerk-Schnittstelle eines virtuellen Controllers.

Tabelle 7- 124 SetIPSuite() - Native C++

Syntax	`ERuntimeErrorCode SetIPSuite(`
	`  UINT32 in_InterfaceID,`
	`  SIPSuite4 in_IPSuite,`
	`  bool in_IsRemanent`
	`);`	

Parameter	• `UINT32 in_InterfaceID:` Die ID der Netzwerk-Schnittstelle. • `SIPSuite4 in_IPSuite:` Die IP-Suite, die der Netzwerk-Schnittstelle zugewiesen werden soll. Die IP-Suite enthält die IP-Adresse, die Subnetzmaske und das Standard-Gateway. Wenn die Kommunikationsschnittstelle "Softbus" ist, dann werden Subnetzmaske und Standard-Gateway ignoriert. • `bool in_IsRemanent:` Wenn `true`, dann wird die IP-Suite nach dem Neustart des virtuellen Controllers gespeichert. Wenn die Kommunikationsschnittstelle "Softbus" ist, dann wird dieses Flag ignoriert.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_DOES_NOT_EXIST`	Es gibt keine Netzwerk-Schnittstelle mit dieser ID.
	`SREC_INVALID_OPERATING_STATE`	Der virtuelle Controller hat den Boot-Prozess noch nicht beendet oder befindet sich bereits in der Shutdown-Phase.

Tabelle 7- 125 SetIPSuite() - .NET (C#)

Syntax	`void SetIPSuite(`
	` UInt32 in_InterfaceID,`
	` SIPSuite4 in_IPSuite,`
	` bool in_IsRemanent );`	

Parameter	• `UInt32 in_InterfaceID:` Die ID der Netzwerk-Schnittstelle. • `SIPSuite4 in_IPSuite:` Wenn die Kommunikationsschnittstelle "Softbus" ist, dann werden Subnetzmaske und Standard-Gateway ignoriert. • `bool in_IsRemanent:` Wenn `true`, dann wird die IP-Suite nach dem Neustart des virtuellen Controllers gespeichert. Wenn die Kommunikationsschnittstelle "Softbus" ist, dann wird dieses Flag ignoriert.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.DoesNotExist`	Es gibt keine Netzwerk-Schnittstelle mit dieser ID.
	`ERuntimeErrorCode.InvalidOperatingState`	Der virtuelle Controller hat den Boot-Prozess noch nicht beendet oder befindet sich bereits in der Shutdown-Phase.

GetStoragePath()

Liefert das vollständige Verzeichnis zurück, in dem die Instanz ihre Daten speichert.

Tabelle 7- 126 GetStoragePath() - Native C++

Syntax	`ERuntimeErrorCode GetStoragePath(`
	`  WCHAR inout_StoragePath[],`
	`  UINT32 in_ArrayLength`
	`);`	

Parameter	• `WCHAR inout_StoragePath[]:` Ein benutzerallokierter Speicher für den Speicherpfad. Die Länge des Arrays soll mindestens so groß sein wie `DSTORAGE_PATH_MAX_LENGTH`. Siehe Datentypen (Seite 316). • `UINT32 in_ArrayLength:` Länge des Arrays (Wide-Character)	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INDEX_OUT_OF_RANGE`	Der Pfad passt nicht in den Speicher.

SetStoragePath()

Setzt den vollständigen Pfad des Verzeichnisses, in dem die Instanz ihre Daten speichert. Dies kann auch eine Netzwerkfreigabe sein.

Setzen Sie den Pfad, bevor Sie die Instanz starten. Eine Änderung des Pfads wirkt sich erst bei einem Neustart des Controllers aus.

Wenn kein Pfad gesetzt ist, gilt die Voreinstellung:
<Eigene Dokumente>\Siemens\Simatic\Simulation\Runtime\Persistence\<Instance Name>.

Tabelle 7- 127 SetStoragePath() - Native C++

Syntax	`ERuntimeErrorCode SetStoragePath(`
	`  WCHAR* in_StoragePath`
	`);`	

Parameter	• `WCHAR* in_StoragePath:` Vollständiger Name des Speicherpfads. Die Länge des Namens muss kürzer sein als `DSTORAGE_PATH_MAX_LENGTH`. Siehe Datentypen (Seite 316).	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INDEX_OUT_OF_RANGE`	Die Länge des Pfads überschreitet das Limit.
	`SREC_WRONG_ARGUMENT`	Der Pfad enthält ungültige Zeichen.

StoragePath { get; set; }

Liefert oder setzt den vollständigen Pfad des Verzeichnisses, in dem die Instanz ihre remanenten Daten speichert. Dies kann auch eine Netzwerkfreigabe sein.

Setzen Sie den Pfad, bevor Sie die Instanz starten. Eine Änderung des Pfads wirkt sich erst bei einem Neustart des Controllers aus.

Wenn kein Pfad gesetzt ist, gilt die Voreinstellung:
<Eigene Dokumente>\Siemens\Simatic\Simulation\Runtime\Persistence\<Instance Name>.

Tabelle 7- 128 StoragePath { get; set; } - .NET (C#)

Syntax	`string StoragePath { get; set; }`	
Parameter	Keine	
Rückgabewerte	`string`: Der konfigurierte Speicherpfad.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Länge des Pfads überschreitet das Limit.
	`ERuntimeErrorCode.WrongArgument`	Der Pfad enthält ungültige Zeichen.

ArchiveStorage()

Das Anwenderprogramm, die Hardware-Konfiguration und die remanenten Daten werden in einer Datei gespeichert, der Virtual SIMATIC Memory Card. `ArchiveStorage()` speichert diese Datei als ZIP-Datei. Die Instanz des virtuellen Controllers muss dazu im Betriebszustand OFF sein.

Tabelle 7- 129 ArchiveStorage() - Native C++

Syntax	`ERuntimeErrorCode ArchiveStorage( WCHAR* in_FullFileName );`	
Parameter	• `WCHAR in_FullFileName:` Der vollständige Pfad zur ZIP-Datei. Der Pfad bezieht sich auf die Verzeichnisse auf dem Computer, auf dem die API aufgerufen wird.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INVALID_OPERATING_STATE`	Die Instanz ist nicht im Betriebszustand OFF.
	`SREC_INVALID_ARCHIVE_PATH`	Der Archiv-Pfad ist ungültig.
	`SREC_CREATE_DIRECTORIES_FAILED`	Das Verzeichnis für die ZIP-Datei konnte nicht erstellt werden.
	`SREC_ARCHIVE_STORAGE_FAILED`	Die ZIP-Datei konnte nicht erstellt werden.
	`SREC_STORAGE_TRANSFER_ERROR`	Fehler beim Netzwerk-Datentransfer. Speicherdaten zwischen Client- und Server-Computern stimmen nicht überein.

Tabelle 7- 130 ArchiveStorage() - .NET (C#)

Syntax	`void ArchiveStorage( string in_FullFileName );`	
Parameter	• `string in_FullFileName:` Der vollständige Pfad zur ZIP-Datei. Der Pfad bezieht sich auf Verzeichnisse des Computers, auf dem die API aufgerufen wird.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InvalidOperatingState`	Die Instanz ist nicht im Betriebszustand OFF.
	`ERuntimeErrorCode.InvalidArchivePath`	Der Archiv-Pfad ist ungültig.

	`ERuntimeErrorCode.CreateDirectoriesFailed`	Das Verzeichnis für die ZIP-Datei konnte nicht erstellt werden.
	`ERuntimeErrorCode.ArchiveStorageNotCreated`	Die ZIP-Datei konnte nicht erstellt werden.
	`ERuntimeErrorCode.StorageTransferError`	Fehler beim Netzwerk-Datentransfer. Speicherdaten zwischen Client- und Server-Computern stimmen nicht überein.

RetrieveStorage()

`RetrieveStorage()` stellt aus der archivierten ZIP-Datei wieder eine Virtual SIMATIC Memory Card her. Der virtuelle Controller muss dazu im Betriebszustand OFF sein.

Tabelle 7- 131 RetrieveStorage() - Native C++

Syntax	`ERuntimeErrorCode RetrieveStorage( WCHAR* in_FullFileName );`	
Parameter	• `WCHAR* in_FullFileName:` Der vollständige Pfad zur ZIP-Datei. Der Pfad bezieht sich auf Verzeichnisse des Computers, auf dem die API aufgerufen wird.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INVALID_OPERATING_STATE`	Die Instanz ist nicht im Betriebszustand OFF.
	`SREC_INVALID_ARCHIVE_PATH`	Der Archiv-Pfad ist ungültig.
	`SREC_DELETE_EXISTING_STORAGE_FAILED`	Der alte Speicher kann nicht gelöscht werden.
	`SREC_RETRIEVE_STORAGE_FAILURE`	Das ZIP-Datei kann nicht entpackt werden.
	`SREC_STORAGE_TRANSFER_ERROR`	Fehler beim Netzwerk-Datentransfer. Speicherdaten zwischen Client- und Server-Computern stimmen nicht überein.

Tabelle 7- 132 RetrieveStorage() - .NET (C#)

Syntax	`void RetrieveStorage( string in_FullFileName );`	
Parameter	• `string in_FullFileName:` Der vollständige Pfad zur ZIP-Datei. Der Pfad bezieht sich auf Verzeichnisse des Computers, auf dem die API aufgerufen wird.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung

	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InvalidOperatingState`	Die Instanz ist nicht im Betriebszustand OFF.
	`ERuntimeErrorCode.InvalidArchivePath`	Der Archiv-Pfad ist ungültig.
	`ERuntimeErrorCode.DeleteExistingStorageFailed`	Der alte Speicher kann nicht gelöscht werden.
	`ERuntimeErrorCode.RetrieveStorageFailure`	Das ZIP-Datei kann nicht entpackt werden.
	`ERuntimeErrorCode.StorageTransferError`	Fehler beim Netzwerk-Datentransfer. Speicherdaten zwischen Client- und Server-Computern stimmen nicht überein.

CleanupStoragePath()

Die Funktion löscht das Verzeichnis mit der Virtual SIMATIC Memory Card einer lokalen Instanz oder einer Remote-Instanz. Dazu prüft die Funktion, ob erforderliche und unzulässige Dateien vorhanden sind. Auch wenn das Verzeichnis fehlt, gilt die Funktion als erfolgreich.

Um sicher zu stellen, dass das korrekte Verzeichnis gelöscht wird, prüft die Funktion, ob die Dateien vorhanden sind, die in der Virtual SIMATIC Memory Card vorhanden sein müssen:

- "../SIMATIC_MC/sim_hwdb.ini"
- "../SIMATIC_MC/SIMATIC.S7S/"
- "../SIMATIC_MC/RData/"

Um das Verzeichnis endgültig zu löschen, sind außerdem nur die folgenden Verzeichnisse mit Dateien erlaubt:

- "../CrashDump/"
- "../Traces/"

Die Instanz muss im Betriebszustand OFF sein ("`PowerOff`").

Tabelle 7- 133 CleanupStoragePath() - Native C++

Syntax	`ERuntimeErrorCode CleanupStoragePath( );`	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INVALID_OPERATING_STATE`	Die Instanz ist nicht im Betriebszustand OFF.
	`SREC_DELETE_EXISTING_STORAGE_FAILED`	Das Verzeichnis mit dem Speicher kann nicht gelöscht werden.
	`SREC_INVALID_STORAGE`	Der Speicher ist ungültig. Er enthält Dateien oder Verzeichnisse, die nicht erlaubt sind.

Tabelle 7- 134 CleanupStoragePath() - .NET (C#)

Syntax	`void CleanupStoragePath( );`	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InvalidOperatingState`	Die Instanz ist nicht im Betriebszustand OFF.
	`ERuntimeErrorCode.DeleteExistingStorageFailed`	Das Verzeichnis mit dem Speicher kann nicht gelöscht werden.
	`SREC_INVALID_STORAGE`	Der Speicher ist ungültig. Er enthält Dateien oder Verzeichnisse, die im Speicher nicht erlaubt sind.

7.6.3 Betriebszustand

PowerOn()

Die Funktion erzeugt den Prozess für die Simulation Runtime Instanz und startet die Firmware des virtuellen Controllers.

Tabelle 7- 135 PowerOn() - Native C++

Syntax	`ERuntimeErrorCode PowerOn();`
	`ERuntimeErrorCode PowerOn(`
	`  UINT32 in_Timeout_ms`
	`);`	

Parameter	• `UINT32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. – Erwartete Betriebszustände, wenn diese Funktion erfolgreich ist: `{ SROS_STOP , SROS_RUN }`	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.

	SREC_ERROR_LOADING_DLL	Die "Siemens.Simatic.Simulation.Runtime.Instance.exe" kann die "Siemens.Simatic.PlcSim.Vplc1500.dll" nicht laden.
	SREC_STORAGE_PATH_ALREADY_IN_USE	Der ausgewählte Pfad für diese Instanz wird bereits von einer anderen Instanz genutzt.
	SREC_NO_STORAGE_PATH_SET	Der Pfad konnte nicht erstellt werden. Eventuell wird die Länge der DSTORAGE_PATH_MAX_LENGTH Zeichen überschritten.
	SREC_WARNING_ALREADY_EXISTS	Warnung: Die Instanz ist gestartet.
	SREC_VIRTUAL_SWITCH_MISCONFIGURED	Der virtuelle Switch ist falsch konfiguriert.
	SREC_INSTANCE_NOT_RUNNING	Der Prozess des virtuellen Controllers läuft nicht mehr.
	SREC_WARNING_UNSUPPORTED_PCAP_DRIVER	Warnung: Der verwendete PCAP-Treiber wird nicht unterstützt. PLCSIM Advanced unterstützt WinPcap V4.1.3. PLCSIM Advanced versucht dennoch hochzufahren, aber es kann Einschränkungen in der TCP/IP-Kommunikation geben.
	SREC_WARNING_TRIAL_MODE_ACTIVE	Warnung: Keine Lizenz verfügbar. Sie können die Instanz ohne Einschränkung mit der Trial License nutzen. Danach wird die Instanz abgeschaltet.
	SREC_WARNING_RUNNING_ON_TIA_PORTAL_TEST_SUITE	Warnung: Es ist keine gültige Lizenz vorhanden, jedoch eine "TIA Portal Test Suite" Lizenz. PLCSIM Advanced startet mit dieser Lizenz. Ein Download aus dem TIA Portal ist möglich, allerdings beendet sich die Instanz ohne Rückmeldung, wenn der Download nicht aus der TIA Portal Test Suite erfolgt ist.
	SREC_NOT_EMPTY	Warnung: Es ist keine gültige Lizenz für PLCSIM Advanced vorhanden, jedoch eine "TIA Portal Test Suite" Lizenz. In diesem Fall wird der Hochlauf aus der Virtual SIMATIC Memory Card nicht unterstützt.

	`SREC_COMMUNICATION_INTERFACE_NOT_AVAILABLE`	**Bei lokaler Kommunikation über Softbus** PLCSIM Advanced kann sich nicht mit dem Softbus verbinden. **Abhilfe** • Versuchen Sie erneut, die Verbindung herzustellen. • Beenden Sie PLCSIM Advanced und das TIA Portal und starten Sie die Anwendungen erneut. • Starten Sie den PC erneut. • Reparieren Sie die PLCSIM Advanced Installation. **Bei TCP/IP-Kommunikation** Auf Ihrem PC ist noch eine Anwendung mit dem Softbus verbunden. **Abhilfe** • Beenden Sie alle SIMATIC Anwendungen, z. B. TIA Portal, WinCC, PLCSIM. • Starten Sie den PC erneut. • Reparieren Sie die PLCSIM Advanced Installation.

Tabelle 7- 136 PowerOn() - .NET (C#)

Syntax	`ERuntimeErrorCode PowerOn();`
	`ERuntimeErrorCode PowerOn(`
	`  UInt32 in_Timeout_ms`
	`);`	

Parameter	• `UInt32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. Erwartete Betriebszustände, wenn diese Funktion erfolgreich ist: `{ EOperatingState.Run, EOperatingState.Stop }`	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.OK`	Die Funktion ist erfolgreich.
	`ERuntimeErrorCode.WarningAlreadyExists`	Warnung: Die Instanz ist gestartet.

	`ERuntimeErrorCode.WarningUnsupportedPcapDriver`	Warnung: Der verwendete PCAP-Treiber wird nicht unterstützt. PLCSIM Advanced unterstützt WinPcap V4.1.3. PLCSIM Advanced versucht dennoch hochzufahren, aber es kann Einschränkungen in der TCP/IP-Kommunikation geben.
	`ERuntimeErrorCode.WarningTrialModeActive`	Warnung: Keine Lizenz verfügbar. Sie können die Instanz ohne Einschränkung mit der Trial License nutzen. Danach wird die Instanz abgeschaltet.
	`ERuntimeErrorCode.WarningRunningOnTiaPortalTestSuite`	Warnung: Es ist keine gültige Lizenz vorhanden, jedoch eine "TIA Portal Test Suite" Lizenz. PLCSIM Advanced startet mit dieser Lizenz. Ein Download aus dem TIA Portal ist möglich, allerdings beendet sich die Instanz ohne Rückmeldung, wenn der Download nicht aus der TIA Portal Test Suite erfolgt ist.
	`ERuntimeErrorCode.NotEmpty`	Warnung: Es ist keine gültige Lizenz für PLCSIM Advanced vorhanden, jedoch eine "TIA Portal Test Suite" Lizenz. In diesem Fall wird der Hochlauf aus der Virtual SIMATIC Memory Card nicht unterstützt.
	`ERuntimeErrorCode.CommunicationInterfaceNotAvailable`	**Bei lokaler Kommunikation über Softbus** PLCSIM Advanced kann sich nicht mit dem Softbus verbinden. **Abhilfe** • Versuchen Sie erneut, die Verbindung herzustellen. • Beenden Sie PLCSIM Advanced und das TIA Portal und starten Sie die Anwendungen erneut. • Starten Sie den PC erneut. • Reparieren Sie die PLCSIM Advanced Installation. **Bei TCP/IP-Kommunikation** Auf Ihrem PC ist noch eine Anwendung mit dem Softbus verbunden. **Abhilfe** • Beenden Sie alle SIMATIC Anwendungen, z. B. TIA Portal, WinCC, PLCSIM. • Starten Sie den PC erneut. • Reparieren Sie die PLCSIM Advanced Installation.

Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`ERuntimeErrorCode.ErrorLoadingDll`	Die "Siemens.Simatic.Simulation. Runtime.Instance.exe" kann die "Siemens.Simatic.PlcSim.Vplc1500.dll" nicht laden.
	`ERuntimeErrorCode.StoragePathAlreadyInUse`	Der ausgewählte Pfad für diese Instanz wird bereits von einer anderen Instanz genutzt.
	`ERuntimeErrorCode.NoStoragePathSet`	Der Pfad konnte nicht erstellt werden. Evtl. wird die Länge der `DSTORAGE_PATH_MAX_LENGTH` Zeichen überschritten.
	`ERuntimeErrorCode.VirtualSwitchMisconfigured`	Der virtuelle Switch ist falsch konfiguriert.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht mehr.

PowerOff()

Fährt die Simulation Runtime herunter und schließt deren Prozess.

Tabelle 7- 137 PowerOff() - Native C++

Syntax	`ERuntimeErrorCode PowerOff();`
	`ERuntimeErrorCode PowerOff(`
	`  UINT32 in_Timeout_ms`
	`);`	

Parameter	• `UINT32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. Erwarteter Betriebszustand, wenn diese Funktion erfolgreich ist: `{ SROS_OFF }`	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

Tabelle 7- 138 PowerOff() - .NET (C#)

Syntax	`void PowerOff();`
	`void PowerOff(`
	`  UInt32 in_Timeout_ms`
	`);`	

Parameter	• `UInt32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. Erwarteter Betriebszustand, wenn diese Funktion erfolgreich ist: `{ EOperatingState.Off }`	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

Run()

Fordert vom virtuellen Controller, in den Betriebszustand RUN zu wechseln.

Tabelle 7- 139 Run() - Native C++

Syntax	`ERuntimeErrorCode Run();`
	`ERuntimeErrorCode Run(`
	`  UINT32 in_Timeout_ms`
	`);`	

Parameter	• `UINT32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. Erwartete Betriebszustände, wenn diese Funktion erfolgreich ist: `{ SROS_STOP , SROS_RUN }`	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

Tabelle 7- 140 Run() - .NET (C#)

Syntax	`void Run();`
	`void Run(`
	`  Uint32 in_Timeout_ms`
	`);`	

Parameter	• `UInt32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. Erwartete Betriebszustände, wenn diese Funktion erfolgreich ist: `{ EOperatingState.Run }`	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

Stop()

Fordert vom virtuellen Controller, in den Betriebszustand STOP zu wechseln.

Tabelle 7- 141 Stop() - Native C++

Syntax	`ERuntimeErrorCode Stop();`
	`ERuntimeErrorCode Stop(`
	`  UINT32 in_Timeout_ms`
	`);`	

Parameter	• `UINT32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. Erwarteter Betriebszustand, wenn diese Funktion erfolgreich ist: `{ SROS_STOP }`	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

Tabelle 7- 142 Stop() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2"><pre>void Stop();
void Stop(
 bool in_IsSynchronous
);</pre></td></tr>
<tr><td>Parameter</td><td colspan="2">• <code>UInt32 in_Timeout_ms:</code>

Ein Timeout-Wert in Millisekunden.
– Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis <code>OnOperatingStateChanged()</code>, um zu erfahren, wenn die Operation durchgeführt wurde.
– Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout.

Erwarteter Betriebszustand, wenn diese Funktion erfolgreich ist:
<code>{ EOperatingState.Stop }</code></td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">Keine</td></tr>
<tr><td rowspan="4">Ausnahmen</td><td colspan="2"><code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code></td></tr>
<tr><td>Runtime Fehlercode</td><td>Bedingung</td></tr>
<tr><td><code>ERuntimeErrorCode.InterfaceRemoved</code></td><td>Die Instanz ist im Runtime Manager nicht registriert.</td></tr>
<tr><td><code>ERuntimeErrorCode.Timeout</code></td><td>Der erwartete Betriebszustand tritt nicht rechtzeitig ein.</td></tr>
<tr><td><code>ERuntimeErrorCode.InstanceNotRunning</code></td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
</table>

GetOperatingState() / OperatingState { get; }

Liefert den Betriebszustand des virtuellen Controllers zurück. Wenn sich der Betriebszustand ändert, wird das Ereignis OnOperatingStateChanged() (Seite 278) ausgelöst. Details zum Betriebszustand siehe Datentypen (Seite 373).

Tabelle 7- 143 GetOperatingState() - Native C++

Syntax	`EOperatingState GetOperatingState();`
Parameter	Keine
Rückgabewerte	• `SROS_INVALID_OPERATING_STATE:` wenn die Funktion fehlschlägt. • `SROS_OFF:` wenn die Simulation Runtime Instanz nicht läuft. • `SROS_BOOTING:` wenn in diesem Zustand `PowerOn()` aufgerufen wurde, aber der virtuelle Controller noch nicht bereit ist, das Anwenderprogramm zu starten. • `SROS_STOP:` wenn der virtuelle Controller in STOP ist. • `SROS_STARTUP:` wenn das Anwenderprogramm gerade von STOP nach RUN wechselt. • `SROS_RUN:` wenn das Anwenderprogramm läuft. • `SROS_FREEZE:` wenn das Anwenderprogramm angehalten wird (Freeze-Status). • `SROS_HOLD:` wenn das Anwenderprogramm beim Erreichen des Haltepunkts in HALT versetzt wird. • `SROS_SHUTTING_DOWN`: wenn `PowerOff()` aufgerufen wurde, aber sich der virtuelle Controller noch in der Shutdown-Phase befindet.

Tabelle 7- 144 OperatingState { get; } - .NET (C#)

Syntax	`EOperatingState OperatingState { get; }`
Parameter	Keine
Rückgabewerte	• `EOperatingState.InvalidOperatingState:` wenn die Funktion fehlschlägt. • `EOperatingState.Off:` wenn die Simulation Runtime Instanz nicht läuft. • `EOperatingState.Booting:` wenn in diesem Zustand `PowerOn()` aufgerufen wurde, aber der virtuelle Controller noch nicht bereit ist, das Anwenderprogramm zu starten. • `EOperatingState.Stop:` wenn der virtuelle Controller in STOP ist. • `EOperatingState.Startup:` wenn das Anwenderprogramm gerade von STOP nach RUN wechselt. • `EOperatingState.Run:` wenn das Anwenderprogramm läuft. • `EOperatingState.Freeze:` wenn das Anwenderprogramm angehalten wird (Freeze-Status). • `EOperatingState.Hold`: wenn das Anwenderprogramm beim Erreichen des Haltepunkts in HALT versetzt wird. • `EOperatingState.ShuttingDown`: wenn `PowerOff()` aufgerufen wurde, aber sich der virtuelle Controller noch in der Shutdown-Phase befindet.

MemoryReset()

Fährt den virtuellen Controller herunter, schließt dessen Prozesse und führt einen Neustart durch.

Tabelle 7- 145 MemoryReset() - Native C++

Syntax	`ERuntimeErrorCode MemoryReset();`
	`ERuntimeErrorCode MemoryReset(`
	`  UINT32 in_Timeout_ms`
	`);`	

Parameter	• `UINT32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. Erwartete Betriebszustände, wenn diese Funktion erfolgreich ist: `{ SROS_STOP, SROS_RUN }`	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.

	SREC_INTERFACE_REMOVED	Die Instanz ist im Runtime Manager nicht registriert.
	SREC_TIMEOUT	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	SREC_INSTANCE_NOT_RUNNING	Der Prozess des virtuellen Controllers läuft nicht.

Tabelle 7- 146 MemoryReset() - .NET (C#)

Syntax	`void MemoryReset();`
	`void MemoryReset(`
	`  UInt32 in_Timeout_ms`
	`);`	

Parameter	• `UInt32 in_Timeout_ms:` Ein Timeout-Wert in Millisekunden. – Wenn kein Timeout-Wert gesetzt ist, dann kehrt die Funktion sofort wieder. Abonnieren Sie das Ereignis `OnOperatingStateChanged()`, um zu erfahren, wenn die Operation durchgeführt wurde. – Wenn der Wert größer als 0 ist (empfohlen ist ein Wert von 60000), kehrt die Funktion wieder, wenn die Operation durchgeführt wurde oder nach einem Timeout. Erwartete Betriebszustände, wenn diese Funktion erfolgreich ist: `{ EOperatingState.Run, EOperatingState.Stop }`	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

7.6.4 Variablentabelle

Hinweis

Elemente mit Datentypen, die der API nicht bekannt sind (`EDataType.Unknown`), werden nicht mit in die Variablentabelle aufgenommen.

UpdateTagList()

Die Funktion liest die Variablen aus dem virtuellen Controller und schreibt sie nach Namen geordnet in den gemeinsamen Speicher.

Wenn die Variable ein Feld oder eine Struktur ist, gibt es mehrfache Einträge.

Bei einer Struktur gibt es einen Eintrag für die Struktur selbst und einen zusätzlichen Eintrag für jedes Strukturelement

Entry_1: "StructName"

Entry_2: "StructName.ElementName_1"

..

Entry_N: "StructName.ElementName_n"

Bei einem Feld, in diesem Beispiel ein zweidimensionales Feld, gibt es einen Eintrag für das Feld selbst und einen zusätzlichen Eintrag für jedes Feldelement.

Entry_1: "ArrayName"

Entry_2: "ArrayName[a,b]" ({a} und {b} entsprechen dem ersten Index der jeweiligen Dimension)

..

Entry_N: "ArrayName[x,y]" ({x} und {y} entsprechen dem letzten Index der jeweiligen Dimension)

Für die Liste ist Speicher für bis zu 500000 Einträge reserviert (nicht PLC-Variablen). Wenn die Liste zu groß wird, liefert die Funktion den Fehler / die Ausnahme "NOT_ENOUGH_MEMORY" zurück.

Wenn es Probleme mit der maximalen Anzahl der Einträge gibt, aber nicht alle Variablen benötigt werden, dann können beim Aktualisieren der Variablentabelle zwei Filter genutzt werden.

Tabelle 7- 147 UpdateTagList() - Native C++

Syntax	`ERuntimeErrorCode UpdateTagList();`
	`ERuntimeErrorCode UpdateTagList(`
	`ETagListDetails in_TagListDetails`
	`);`
	`ERuntimeErrorCode UpdateTagList(`
	`ETagListDetails in_TagListDetails,`
	`bool in_IsHMIVisibleOnly`
	`);`
	`ERuntimeErrorCode UpdateTagList(`
	`ETagListDetails in_TagListDetails,`
	`bool in_IsHMIVisibleOnly,`
	`WCHAR* in_DataBlockFilterList`
	`);`	

Parameter	• `ETagListDetails in_TagListDetails:` Jede Kombination der folgenden vier Bereiche: **IO**: Eingänge und Ausgänge **M**: Merker **CT**: Zähler und Zeiten **DB**: Datenbausteine Die Voreinstellung ist **IOMCTDB**. **Beispiel: IOM** liest nur die Variablen aus der Area Eingänge / Ausgänge und Merker. • `bool in_IsHMIVisibleOnly:` Wenn `true`, werden nur Variablen gelesen, die mit "HMI Visible" markiert sind. Die Voreinstellung ist `true`. • `WCHAR* in_DataBlockFilterList:` Ein String, der die Namen aller Datenbausteine enthält, die im Variablentabellen-Speicher verfügbar sein sollen. Der String muss in Anführungszeichen stehen. **Beispiel:** ""\"DB_1\", \"DB_2\" \"DB_3\"|\"DB_4\"\"DB_5\"" Alle Zeichen innerhalb der Anführungszeichen werden als DB-Name interpretiert. Wenn der Datenbaustein im PLC-Programm nicht existiert, wird er dem Variablentabellen-Speicher nicht hinzugefügt, dabei wird kein Fehler ausgelöst. Damit diese Liste berücksichtigt wird, muss `in_DataBlockFilterList` ungleich NULL sein und `in_TagListDetails` muss "DB" enthalten.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_NOT_ENOUGH_MEMORY`	Es werden mehr als 500000 Einträge angefordert.

	SREC_WARNING_ALREADY_EXISTS	Die Variablentabelle ist aktuell.
	SREC_WRONG_ARGUMENT	Die Syntax von in_DataBlockFilter-List ist ungültig. Die Liste muss 3 Zeichen lang sein, das erste und letzte Zeichen muss jeweils ein Anführungszeichen sein.

Tabelle 7- 148 UpdateTagList() - .NET (C#)

Syntax	void UpdateTagList(); void UpdateTagList(ETagListDetails in_TagListDetails); void UpdateTagList(ETagListDetails in_TagListDetails, bool in_IsHMIVisibleOnly); ERuntimeErrorCode UpdateTagList(ETagListDetails in_TagListDetails, bool in_IsHMIVisibleOnly, string in_DataBlockFilterList);	
Parameter	• ETagListDetails in_TagListDetails: Jede Kombination der folgenden vier Bereiche: **IO**: Eingänge und Ausgänge **M**: Merker **CT**: Zähler und Zeiten **DB**: Datenbausteine Die Voreinstellung ist **IOMCTDB**. **Beispiel: IOM** liest nur die Variablen aus der Area Eingänge / Ausgänge und Merker. • bool in_IsHMIVisibleOnly: Wenn true, werden nur Variablen gelesen, die mit "HMI Visible" markiert sind. Die Voreinstellung ist true. • string in_DataBlockFilterList: Ein String, der die Namen aller Datenbausteine enthält, die im Variablentabellen-Speicher verfügbar sein sollen. Der String muss in Anführungszeichen stehen. **Beispiel:** ""\"DB_1\", \"DB_2\" \"DB_3\"|\"DB_4\"\"DB_5\"" Alle Zeichen innerhalb der Anführungszeichen werden als DB-Name interpretiert. Wenn der Datenbaustein im PLC-Programm nicht existiert, wird er dem Variablentabellen-Speicher nicht hinzugefügt, dabei wird kein Fehler ausgelöst. Damit diese Liste berücksichtigt wird, muss in_DataBlockFilterList ungleich NULL sein und in_TagListDetails muss "DB" enthalten.	
Rückgabewerte	Keine	
Ausnahmen	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime Fehlercode	Bedingung
	ERuntimeErrorCode.InterfaceRemoved	Die Instanz ist im Runtime Manager nicht registriert.

	ERuntimeErrorCode.Timeout	Die Funktion kehrt nicht rechtzeitig wieder.
	ERuntimeErrorCode.InstanceNotRunning	Der Prozess des virtuellen Controllers läuft nicht.
	ERuntimeErrorCode.NotEnoughMemory	Es werden mehr als 500000 Einträge angefordert.
	ERuntimeErrorCode.WrongArgument	Die Syntax von `in_DataBlockFilterList` ist ungültig. Die Liste muss 3 Zeichen lang sein, das erste und letzte Zeichen muss jeweils ein Anführungszeichen sein.

GetTagListStatus()

Liefert den aktuellen Update-Status des Variablentabellen-Speichers zurück.

"`inout_TagListDetails`" ist `NONE`, wenn die Tabelle aktualisiert werden muss.

Tabelle 7- 149 GetTagListStatus() - Native C++

Syntax	`ERuntimeErrorCode GetTagListStatus(`
	` ETagListDetails* out_TagListDetails,`
	` bool* out_IsHMIVisibleOnly`
	`);`	

Parameter	• `ETagListDetails out_TagListDetails:` Status der Variablentabellen-Details. `SRTLD_NONE`, wenn ein Update der Tabelle erforderlich ist. • `bool out_IsHMIVisibleOnly:` Wenn `true`, sind nur Variablen in der Tabelle verfügbar, die mit "HMI Visible" markiert sind.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	SREC_OK	Die Funktion ist erfolgreich.
	SREC_INTERFACE_REMOVED	Die Instanz ist im Runtime Manager nicht registriert.
	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 150 GetTagListStatus() - .NET (C#)

Syntax	`void GetTagListStatus(`
	` out ETagListDetails out_TagListDetails,`
	` out bool out_IsHMIVisibleOnly`
	`);`

Parameter	• `out ETagListDetails out_TagListDetails:` Status der Variablentabellen-Details. `ETagListDetails.None`, wenn ein Update der Tabelle erforderlich ist. • `out bool out_IsHMIVisibleOnly:` Wenn `true`, sind nur Variablen in der Tabelle verfügbar, die mit "HMI Visible" markiert sind.

Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

GetTagInfoCount()

Liefert die Anzahl der Einträge im Variablentabellen-Speicher zurück. Wenn die Funktion fehlschlägt, ist der Rückgabewert 0.

Tabelle 7- 151 GetTagInfoCount() - Native C++

Syntax	`UINT32 GetTagInfoCount();`
Parameter	Keine
Rückgabewerte	Anzahl der Einträge im Variablentabellen-Speicher.

GetTagInfos() / TagInfos { get; }

Liefert eine Liste aller Variablen zurück.

Tabelle 7- 152 GetTagInfos() - Native C++

Syntax	`ERuntimeErrorCode GetTagInfos(`
	`  UINT32 in_BufferLength,`
	`  STagInfo* inout_TagInfos,`
	`  UINT32* out_TagCount`
	`);`	

Parameter	• `UINT32 in_BufferLength:` Die Anzahl der Elemente, die der Speicher aufnehmen kann. • `STagInfo* inout_TagInfos:` Der benutzerallokierte Speicher, der die Variablen aufnimmt. • `UINT32* out_TagCount:` Liefert die Anzahl der Variablen zurück, die in den Speicher geschrieben wurden.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Die Elemente passen nicht in den Speicher.

Tabelle 7- 153 TagInfos { get; } - .NET (C#)

Syntax	`STagInfo[] TagInfos { get; }`	
Parameter	Keine	
Rückgabewerte	Ein Feld, das alle verfügbaren Einträge des Speichers enthält.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

CreateConfigurationFile()

Schreibt alle Einträge aus der Variablentabelle in eine XML-Datei.

Tabelle 7- 154 CreateConfigurationFile() - Native C++

Syntax	`ERuntimeErrorCode CreateConfigurationFile( WCHAR* in_FullFileName );`	
Parameter	• `WCHAR* in_FullFileName:` Vollständiger Dateiname der XML-Datei: <Pfad> + <Dateiname> + <Dateiendung>.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Dateiname ist ungültig.

Tabelle 7- 155 CreateConfigurationFile() - .NET (C#)

Syntax	`void CreateConfigurationFile( string in_FullFileName );`	
Parameter	Keine	
Rückgabewerte	• `string in_FullFileName:` Dateiname der XML-Datei, in die geschrieben wird: <Pfad> + <Dateiname> + <Dateiendung>.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Dateiname ist ungültig.

7.6.5 I/O-Zugriff

7.6.5.1 Synchronisieren von Eingängen und Ausgängen

Beschreibung

In PLCSIM Advanced kann der komplette Umfang des Eingangs- und Ausgangsbereichs genutzt werden (siehe GetAreaSize/AreaSize (Seite 185)). Dies ist auch dann möglich, wenn kein IO-Modul konfiguriert ist.

Eingänge und Ausgänge, die über konfigurierte IO-Module definiert sind, werden zur festgelegten Aktualisierung des Teilprozessabbilds (TPA) synchronisiert.

Eingänge und Ausgänge, die keinem IO-Modul zugeordnet sind, werden am Zykluskontrollpunkt synchronisiert.

Beachten Sie beim Synchronisieren dieser Eingänge und Ausgänge Folgendes:

- Eingänge können nur als Eingänge genutzt werden.

  Sie können Werte über die API schreiben, aber Werte, die über das Anwenderprogramm (TIA Portal) geschrieben werden, sind nicht sichtbar in der API.

- Ausgänge können sowohl als Ausgang als auch als Eingang genutzt werden.

  Sie können Werte über die API und über die CPU / das Anwenderprogramm (TIA Portal) schreiben. Wenn API und Anwenderprogramm auf den gleichen Bereich schreiben, dann werden die Werte aus der API die Werte aus dem Anwenderprogramm überschreiben.

7.6.5.2 I/O-Zugriff über Adresse - Lesen

InputArea { get; }, MarkerArea { get; }, OutputArea { get; }

Liefert eine Schnittstelle zurück, die Sie nutzen, um die .NET-Funktionen in diesem Kapitel aufzurufen.

Tabelle 7- 156 InputArea { get; } MarkerArea { get; } OutputArea { get; } - .NET (C#)

Syntax	`IIOArea InputArea { get; }`
	`IIOArea MarkerArea { get; }`
	`IIOArea OutputArea { get; }`

Parameter	Keine
Rückgabewerte	`IIOArea`: Die Schnittstelle, die genutzt wird, um die Funktionen "I/O-Zugriff über Adresse" aufzurufen.

GetAreaSize() / AreaSize { get; }

Liefert die Größe der Area in Bytes zurück.

Tabelle 7- 157 GetAreaSize() - Native C++

Syntax	`UINT32 GetAreaSize(`
	`  EArea in_Area`
	`);`

Parameter	• `EArea in_Area:` Die Area, von der Sie die Größe erhalten möchten. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372).
Rückgabewerte	`UINT32`: Größe der Area in Bytes. Wenn die Funktion erfolgreich war, ist der Wert ungleich 0.

Tabelle 7- 158 AreaSize { get; } - .NET (C#)

Syntax	`UInt32 InputArea.AreaSize { get; }`
	`UInt32 MarkerArea.AreaSize { get; }`
	`UInt32 OutputArea.AreaSize { get; }`

Parameter	Keine
Rückgabewerte	`UInt32`: Größe der Area in Bytes. Wenn die Funktion erfolgreich war, ist der Wert ungleich 0.

ReadBit()

Liest ein einzelnes Bit aus der Area.

Hinweis

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Nutzen Sie daher den Zugriff über den Variablennamen (Seite 201) und nicht über die Adressbereiche.

Tabelle 7- 159 ReadBit() - Native C++

Syntax	`ERuntimeErrorCode ReadBit(`
	` EArea in_Area,`
	` UINT32 in_Offset,`
	` UINT8 in_Bit,`
	` bool* out_Value`
	`);`	

Parameter	• `EArea in_Area:` Die Area , von der gelesen werden soll. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372). • `UINT32 in_Offset:` Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, den `GetAreaSize()` zurückgibt. • `UINT8 in_Bit:` Der Bit-Offset innerhalb des Bytes. Der Wert muss zwischen 0 und 7 sein. • `bool* out_Value:` Gibt den Bitwert zurück.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Offset oder Bits sind ungültig.
	`SREC_WRONG_ARGUMENT`	Die Area ist ungültig.

Tabelle 7- 160 ReadBit() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2">

```
bool InputArea.ReadBit(
 UInt32 in_Offset,
 Byte in_Bit
);
bool MarkerArea.ReadBit(
 UInt32 in_Offset,
 Byte in_Bit
);
bool OutputArea.ReadBit(
 UInt32 in_Offset,
 Byte in_Bit
);
```

</td></tr>
<tr><td>Parameter</td><td colspan="2">

- `UInt32 in_Offset:`

  Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, den `AreaSize` zurückgibt.

- `Byte in_Bit:`

  Der Bit-Offset innerhalb des Bytes. Der Wert muss zwischen 0 und 7 sein.

</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">`bool:` Bitwert</td></tr>
<tr><td rowspan="5">Ausnahmen</td><td colspan="2">`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`</td></tr>
<tr><td>Runtime Fehlercode</td><td>Bedingung</td></tr>
<tr><td>`ERuntimeErrorCode.InterfaceRemoved`</td><td>Die Instanz ist im Runtime Manager nicht registriert.</td></tr>
<tr><td>`ERuntimeErrorCode.Timeout`</td><td>Die Funktion kehrt nicht rechtzeitig wieder.</td></tr>
<tr><td>`ERuntimeErrorCode.InstanceNotRunning`</td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td></td><td>`ERuntimeErrorCode.IndexOutOfRange`</td><td>Offset oder Bits sind ungültig.</td></tr>
</table>

ReadByte()

Liest ein einzelnes Byte aus der Area.

Hinweis

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 161 ReadByte() - Native C++

Syntax	`ERuntimeErrorCode ReadByte( EArea in_Area, UINT32 in_Offset, BYTE* out_Value);`	
Parameter	• `EArea in_Area:` Die Area, von der Sie lesen möchten. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372). • `UINT32 in_Offset`: Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, der von `GetAreaSize()` zurückgegeben wird. • `BYTE* out_Value:` Gibt den Bytewert zurück.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Offset ist ungültig.
	`SREC_WRONG_ARGUMENT`	Die Area ist ungültig.

Tabelle 7- 162 ReadByte() - .NET (C#)

Syntax	`Byte InputArea.ReadByte( UInt32 in_Offset );`
	`Byte MarkerArea.ReadByte( UInt32 in_Offset );`
	`Byte OutputArea.ReadByte( UInt32 in_Offset );`	

Parameter	• `UInt32 in_Offset:` Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, der von `AreaSize` zurückgegeben wird.	
Rückgabewerte	`Byte:` Bytewert.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung

	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Offset ist ungültig.

ReadBytes()

Liest ein Byte-Array aus der Area.

Hinweis

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 163 ReadByte() - Native C++

Syntax	`ERuntimeErrorCode ReadBytes(`
	` EArea in_Area,`
	` UINT32 in_Offset,`
	` UINT32 in_BytesToRead,`
	` UINT32* out_BytesRead,`
	` BYTE inout_Values[]`
	`);`	

Parameter	• `EArea in_Area:` Die Area, von der Sie lesen möchten. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372). • `UINT32 in_Offset`: Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, der von `GetAreaSize()` zurückgegeben wird. • `UINT32 in_BytesToRead:` Enthält die Größe des Wertespeichers. • `UINT32* out_BytesRead:` Enthält die Anzahl der Bytes, die gerade in den Wertespeicher geschrieben wurden. • `BYTE inout_Values[]:` Der Speicher für die Bytes, die aus der Area gelesen werden.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Byte konnte gelesen werden.
	`SREC_WRONG_ARGUMENT`	Die Area ist ungültig.

Tabelle 7- 164 ReadBytes() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2">

```
Byte[] InputArea.ReadBytes(
 UInt32 in_Offset,
 UInt32 in_BytesToRead
);
Byte[] MarkerArea.ReadBytes(
 UInt32 in_Offset,
 UInt32 in_BytesToRead
);
Byte[] OutputArea.ReadBytes(
 UInt32 in_Offset,
 UInt32 in_BytesToRead
);
```

</td></tr>
<tr><td>Parameter</td><td colspan="2">

- `UInt32 in_Offset:`

  Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, der von `AreaSize` zurückgegeben wird.

- `UInt32 in_BytesToRead:`

  Die Anzahl der Bytes, die gelesen werden.

</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">`Byte[]`: Die gelesenen Bytes.</td></tr>
<tr><td rowspan="5">Ausnahmen</td><td colspan="2">`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`</td></tr>
<tr><td>Runtime Fehlercode</td><td>Bedingung</td></tr>
<tr><td>`ERuntimeErrorCode.InterfaceRemoved`</td><td>Die Instanz ist im Runtime Manager nicht registriert.</td></tr>
<tr><td>`ERuntimeErrorCode.Timeout`</td><td>Die Funktion kehrt nicht rechtzeitig wieder.</td></tr>
<tr><td>`ERuntimeErrorCode.InstanceNotRunning`</td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td></td><td>`ERuntimeErrorCode.IndexOutOfRange`</td><td>Der Offset liegt außerhalb der Area-Größe. Kein Byte konnte gelesen werden.</td></tr>
</table>

ReadSignals()

Strukturen und Felder können durch Signallisten nachgebildet und dann über die Funktion `ReadSignals()` gelesen werden.

Die Funktion berücksichtigt auch die Byte-Reihenfolge (Endianness).

Es werden nur primitive Datentyp-Signale unterstützt, aber die Funktion ist nicht typsicher.

Hinweis

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Nutzen Sie daher den Zugriff über den Variablennamen (Seite 201) und nicht über die Adressbereiche.

Tabelle 7- 165 ReadSignals() - Native C++

Syntax	`ERuntimeErrorCode ReadSignals(`
	` EArea in_Area,`
	` SDataValueByAddress* inout_Signals,`
	` UINT32 in_SignalCount`
	`);`
	`ERuntimeErrorCode ReadSignals(`
	` EArea in_Area,`
	` SDataValueByAddressWithCheck* inout_Signals,`
	` UINT32 in_SignalCount,`
	` bool* out_SignalsHaveChanged`
	`);`	

Parameter	• `EArea in_Area:` Die Area, von der Sie lesen möchten. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372). • `SDataValueByAddress* inout_Signals:` Die Signalliste, die gelesen wird. Das Ergebnis wird in der Struktur gespeichert. • `SDataValueByAddressWithCheck* inout_Signals:` Die Signalliste, die gelesen wird. Das Ergebnis wird in der Struktur gespeichert. "`ValueHasChanged`" wird auf `true` gesetzt, wenn sich der Wert des Signals seit dem vorhergehenden Aufruf geändert hat. • `UINT32 in_SignalCount:` Die Anzahl der Signale in der Liste. • `bool* out_SignalsHaveChanged:` Gibt `true` zurück, wenn sich der Wert von mindestens einem Signal seit dem vorhergehenden Aufruf geändert hat.	
Signalfehler	Fehlercode	Bedingung
	`SREC_OK`	Die Signaloperation ist erfolgreich.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).

Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_WRONG_ARGUMENT`	Die Area ist ungültig.
	`SREC_SIGNAL_CONFIGURATION_ERROR`	Mindestens ein Signalfehler ist in der Liste.

Tabelle 7- 166 ReadSignals() - .NET (C#)

Syntax		

	`void ReadSignals(`
	` ref SDataValueByAddress[] inout_Signals`
	`);`
	`void ReadSignals(`
	` ref SDataValueByAddressWithCheck[] inout_Signals`
	` out bool out_SignalsHaveChanged);`
	`);`	

Parameter	• `ref SDataValueByAddress[] inout_Signals:` Die Signalliste, die gelesen wird. • `ref SDataValueByAddressWithCheck[] inout_Signals:` Die Signalliste, die gelesen wird. Das Ergebnis wird in der Struktur gespeichert. "`ValueHasChanged`" wird auf `true` gesetzt, wenn sich der Wert des Signals seit dem vorhergehenden Aufruf geändert hat. • `out bool out_SignalsHaveChanged:` Gibt `true` zurück, wenn sich der Wert von mindestens einem Signal seit dem vorhergehenden Aufruf geändert hat.	
Rückgabewerte	Keine	
Signalfehler	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Ok`	Die Signaloperation ist erfolgreich.
	`ERuntimeErrorCode.IndexOutOfRange`	Offset oder Bits sind ungültig.
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.SignalConfigurationError`	Mindestens ein Signalfehler ist in der Liste.

7.6.5.3 I/O-Zugriff über Adresse - Schreiben

WriteBit()

Schreibt ein einzelnes Bit in die Area.

Hinweis
Daten können überschrieben werden

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Nutzen Sie daher den Zugriff über den Variablennamen (Seite 224) und nicht über die Adressbereiche.

Tabelle 7- 167 WriteBit() - Native C++

Syntax	`ERuntimeErrorCode WriteBit(`
	` EArea in_Area,`
	` UINT32 in_Offset,`
	` UINT8 in_Bit,`
	` bool in_Value`
	`);`	

Parameter	• `EArea in_Area:` Die Area, in die geschrieben werden soll. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372). • `UINT32 in_Offset:` Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, der von `GetAreaSize()` zurückgegeben wird. • `UINT8 in_Bit:` Der Bit-Offset innerhalb des Bytes. Der Wert muss zwischen 0 und 7 sein. • `bool in_Value:` Bitwert.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Offset oder Bits sind ungültig.
	`SREC_WRONG_ARGUMENT`	Area ist ungültig.

Tabelle 7- 168 WriteBit() - .NET (C#)

Syntax	`void InputArea WriteBit(`
	` UInt32 in_Offset,`
	` Byte in_Bit,`
	` bool in_Value`
	`);`
	`void MarkerArea WriteBit(`
	` UInt32 in_Offset,`
	` Byte in_Bit,`
	` bool in_Value`
	`);`
	`void OutputArea WriteBit(`
	` UInt32 in_Offset,`
	` Byte in_Bit,`
	` bool in_Value`
	`);`	

Parameter	• `UInt32 in_Offset:` Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, der von `AreaSize` zurückgegeben wird. • `Byte in_Bit:` Der Bit-Offset innerhalb des Bytes. Der Wert muss zwischen 0 und 7 sein. • `bool in_Value:` Bitwert.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Offset oder Bits sind ungültig.

WriteByte()

Schreibt ein einzelnes Byte in die Area.

Hinweis

Daten können überschrieben werden

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Nutzen Sie daher den Zugriff über den Variablennamen (Seite 224) und nicht über die Adressbereiche.

Tabelle 7- 169 WriteByte() - Native C++

Syntax	`ERuntimeErrorCode WriteByte(`
	`  EArea in_Area,`
	`  UINT32 in_Offset,`
	`  BYTE in_Value);`	

Parameter	• `EArea in_Area:` Die Area, in die geschrieben werden soll. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372). • `UINT32 in_Offset:` Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, der von `GetAreaSize()` zurückgegeben wird. • `BYTE in_Value:` Bytewert.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Offset ist ungültig.
	`SREC_WRONG_ARGUMENT`	Area ist ungültig.

Tabelle 7- 170 WriteByte() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2">

```
void InputArea.WriteByte(
 UInt32 in_Offset,
 Byte in_Value
);
void MarkerArea.WriteByte(
 UInt32 in_Offset,
 Byte in_Value
);
void OutputArea.WriteByte(
 UInt32 in_Offset,
 Byte in_Value
);
```

</td></tr>
<tr><td>Parameter</td><td colspan="2">

- `UINT32 in_Offset:`

  Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, der von `AreaSize` zurückgegeben wird.

- `BYTE in_Value:`

  Bytewert.

</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">Keine</td></tr>
<tr><td rowspan="5">Ausnahmen</td><td colspan="2">`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`</td></tr>
<tr><td>Runtime Fehlercode</td><td>Bedingung</td></tr>
<tr><td>`ERuntimeErrorCode.InterfaceRemoved`</td><td>Die Instanz ist im Runtime Manager nicht registriert.</td></tr>
<tr><td>`ERuntimeErrorCode.Timeout`</td><td>Die Funktion kehrt nicht rechtzeitig wieder.</td></tr>
<tr><td>`ERuntimeErrorCode.InstanceNotRunning`</td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td></td><td>`ERuntimeErrorCode.IndexOutOfRange`</td><td>Offset ist ungültig.</td></tr>
</table>

WriteBytes()

Schreibt ein Byte-Array in die Area.

Hinweis

Daten können überschrieben werden

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Achten Sie besonders darauf, nicht auf Bytes zu schreiben, die zu anderen Applikationen gehören oder die interne Daten enthalten, z. B. Qualifier Bits für fehlersichere Peripheriemodule.

Nutzen Sie daher den Zugriff über den Variablennamen (Seite 224) und nicht über die Adressbereiche.

Tabelle 7- 171 WriteBytes() - Native C++

Syntax	`ERuntimeErrorCode WriteBytes(`
	` EArea in_Area,`
	` UINT32 in_Offset,`
	` UINT32 in_BytesToWrite,`
	` UINT32* out_BytesWritten,`
	` BYTE in_Values[])`
	`;`	

Parameter	• `EArea in_Area:` Die Area, in die geschrieben werden soll. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372). • `UINT32 in_Offset:` Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, den `GetAreaSize()` zurückgibt. • `UINT32 in_BytesToWrite:` Enthält die Größe des Arraywerts, der geschrieben wird. • `UINT32* out_BytesWritten:` Enthält die Anzahl der Bytes, die gerade geschrieben wurden. • `BYTE in_Values[]:` Byte-Array, das in die Area geschrieben werden soll.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Byte konnte geschrieben werden.
	`SREC_WRONG_ARGUMENT`	Die Area ist ungültig.

Tabelle 7- 172 WriteBytes() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2">
<pre>
UInt32 InputArea.WriteBytes(
 UInt32 in_Offset,
 Byte[] in_Values
);
UInt32 InputArea.WriteBytes(
 UInt32 in_Offset,
 UInt32 in_BytesToWrite,
 Byte[] in_Values

);
UInt32 MarkerArea.WriteBytes(
 UInt32 in_Offset,
 Byte[] in_Values
);
UInt32 MarkerArea.WriteBytes(
 UInt32 in_Offset,
 UInt32 in_BytesToWrite,
 Byte[] in_Values
);
UInt32 OutputArea.WriteBytes(
 UInt32 in_Offset,
 Byte[] in_Values
);
UInt32 OutputArea.WriteBytes(
 UInt32 in_Offset,
 UInt32 in_BytesToWrite,
 Byte[] in_Values
);
</pre>
</td></tr>
<tr><td>Parameter</td><td colspan="2">

<code>UINT32 in_Offset:</code>
Der Byte-Offset innerhalb der Area. Der Wert muss zwischen 0 und dem Wert sein, den <code>AreaSize</code> zurückgibt.
<code>UInt32 in_BytesToWrite:</code>
Enthält die Anzahl der Bytes, die geschrieben werden. Der Wert muss zwischen 1 und der Größe des Arraywerts sein.
<code>BYTE in_Value:</code>
Bytewert.

</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2"><code>UInt32:</code> Enthält die Anzahl der Bytes, die gerade geschrieben wurden.</td></tr>
<tr><td rowspan="5">Ausnahmen</td><td colspan="2"><code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code></td></tr>
<tr><td>Runtime Fehlercode</td><td>Bedingung</td></tr>
<tr><td><code>ERuntimeErrorCode.InterfaceRemoved</code></td><td>Die Instanz ist im Runtime Manager nicht registriert.</td></tr>
<tr><td><code>ERuntimeErrorCode.Timeout</code></td><td>Die Funktion kehrt nicht rechtzeitig wieder.</td></tr>
<tr><td><code>ERuntimeErrorCode.InstanceNotRunning</code></td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td><code>ERuntimeErrorCode.IndexOutOfRange</code></td><td>Der Offset liegt außerhalb der Area-Größe. Kein Byte konnte geschrieben werden.</td></tr>
</table>

WriteSignals()

Schreibt mehrere Signale innerhalb eines API-Aufrufs. Die Funktion berücksichtigt auch die Byte-Reihenfolge (Endianness).

Die Funktion unterstützt nur primitive Datentyp-Signale, sie ist aber nicht typsicher.

Hinweis

Daten können überschrieben werden

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Nutzen Sie daher den Zugriff über den Variablennamen (Seite 224) und nicht über die Adressbereiche.

Tabelle 7- 173 WriteSignals() - Native C++

Syntax	`ERuntimeErrorCode WriteSignals(`
	` EArea in_Area,`
	` SDataValueByAddress* in_Signals,`
	` UINT32 in_SignalCount`
	`);`	

Parameter	• `EArea in_Area:` Die Area, in die geschrieben wird. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 372). • `SDataValueByAddress* inout_Signals:` Die Signalliste, die geschrieben wird. • `UINT32 in_SignalCount:` Die Anzahl der Signale in der Liste.	
Signalfehler	Fehlercode	Bedingung
	`SREC_OK`	Die Signaloperation ist erfolgreich.
	`SREC_INDEX_OUT_OF_RANGE`	Offset oder Bits sind ungültig.
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_SIGNAL_CONFIGURATION_ERROR`	Mindestens ein Signalfehler ist in der Liste.
	`SREC_WRONG_ARGUMENT`	Die Area ist ungültig.

Tabelle 7- 174 WriteSignals() - .NET (C#)

Syntax	`void InputArea.WriteSignals( SDataValueByAddress[] in_Signals );`
	`void MarkerArea.WriteSignals( SDataValueByAddress[] in_Signals );`
	`void OutputArea.WriteSignals( SDataValueByAddress[] in_Signals );`	

Parameter	• `SDataValueByAddress[] in_Signals:` Die Signalliste, die geschrieben wird.	
Rückgabewerte	Keine	
Signalfehler	Fehlercode	Bedingung
	`ERuntimeErrorCode.Ok`	Die Signaloperation ist erfolgreich.
	`ERuntimeError-Code.IndexOutOfRange`	Offset oder Bits sind ungültig.
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.SignalConfigurationError`	Mindestens ein Signalfehler ist in der Liste.

7.6.5.4 I/O-Zugriff über Variablenname - Lesen

Einzelzugriffe auf IO-Daten sind dazu geeignet, Werte, die nicht regelmäßig aktualisiert werden, auf einer grafischen Oberfläche (GUI) anzuzeigen und zu schreiben.

Hinweis

Um einen regelmäßigen Signalaustausch zu simulieren, erstellen Sie einmal eine Signalliste für jeden Satz an Signalen. Nutzen Sie diese Signalliste für alle weiteren Zugriffe. Erstellen Sie eine neue Liste, sobald sich der Satz an Signalen ändert.

Verwenden Sie für die Signallisten die Funktionen `ReadSignals()` und `WriteSignals()`.

Read()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 175 Read() - Native C++

Syntax	`ERuntimeErrorCode Read(`
	` WCHAR* in_Tag,`
	` SDataValue* inout_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `SDataValue* inout_Value:` Enthält den Wert und den erwarteten Typ der PLC-Variablen. Wenn der erwartete Typ `UNSPECIFIC` ist, dann wird er auf den gespeicherten Typ gesetzt, wenn die Funktion erfolgreich war. Der Typ `STRUCT` wird nicht unterstützt. Strukturen und Felder können durch Signallisten nachgebildet und dann über `ReadSignals()` gelesen werden.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 176 Read() - .NET (C#)

Syntax	`SDataValue Read(`
	` string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`SDataValue:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 381).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadBool()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 177 ReadBool() - Native C++

Syntax	`ERuntimeErrorCode ReadBool(`
	` WCHAR* in_Tag,`
	` bool* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `bool* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

	SREC_INDEX_OUT_OF_RANGE	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 178 ReadBool() - .NET (C#)

Syntax	`bool ReadBool(`
	`  string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`bool`: Enthält den Wert der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadInt8()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 179 ReadInt8() - Native C++

Syntax	`ERuntimeErrorCode ReadInt8(`
	`  WCHAR* in_Tag,`
	`  INT8* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `INT8* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 180 ReadInt8() - .NET (C#)

Syntax	`Int8 ReadInt8(`
	`  string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`Int8:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

	`ERuntimeErrorCode.InstanceNot-Running`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.In-dexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadInt16()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 181 ReadInt16() - Native C++

Syntax	`ERuntimeErrorCode ReadInt16(`
	`  WCHAR* in_Tag,`
	`  INT16* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `INT16* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 182 ReadInt16() - .NET (C#)

Syntax	`Int16 ReadInt16(`
	` string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`Int16:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadInt32()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 183 ReadInt32() - Native C++

Syntax	`ERuntimeErrorCode ReadInt32(`
	` WCHAR* in_Tag,`
	` INT32* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `INT32* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

	SREC_INDEX_OUT_OF_RANGE	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 184 ReadInt32() - .NET (C#)

Syntax	`Int32 ReadInt32(`
	`  string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`Int32:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadInt64()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 185 ReadInt64() - Native C++

Syntax	`ERuntimeErrorCode ReadInt64(`
	`  WCHAR* in_Tag,`
	`  INT64* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `INT64* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 186 ReadInt64() - .NET (C#)

Syntax	`Int64 ReadInt64(`
	`  string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`Int64:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

	`ERuntimeErrorCode.InstanceNot-Running`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.In-dexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadUInt8()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 187 ReadUInt8() - Native C++

Syntax	`ERuntimeErrorCode ReadUInt8(`
	`  WCHAR* in_Tag,`
	`  UINT8* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `UINT8* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 188 ReadUInt8() - .NET (C#)

Syntax	`UInt8 ReadUInt8(`
	` string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`UInt8:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadUInt16()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 189 ReadUInt16() - Native C++

Syntax	`ERuntimeErrorCode ReadUInt16(`
	` WCHAR* in_Tag,`
	` UINT16* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `UINT16* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

	SREC_INDEX_OUT_OF_RANGE	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 190 ReadUInt16() - .NET (C#)

Syntax	UInt16 ReadUInt16(string in_Tag)	
Parameter	• string in_Tag: Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	UInt16: Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime Fehlercode	Bedingung
	ERuntimeErrorCode.InterfaceRemoved	Die Instanz ist im Runtime Manager nicht registriert.
	ERuntimeErrorCode.Timeout	Die Funktion kehrt nicht rechtzeitig wieder.
	ERuntimeErrorCode.InstanceNotRunning	Der Prozess des virtuellen Controllers läuft nicht.
	ERuntimeErrorCode.IndexOutOfRange	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	ERuntimeErrorCode.DoesNotExist	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	ERuntimeErrorCode.NotSupported	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	ERuntimeErrorCode.TypeMismatch	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	ERuntimeErrorCode.NotUpToData	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadUInt32()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 191 ReadUInt32() - Native C++

Syntax	`ERuntimeErrorCode ReadUInt32(`
	`  WCHAR* in_Tag,`
	`  UINT32* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `UINT32* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 192 ReadUInt32() - .NET (C#)

Syntax	`UInt32 ReadUInt32(`
	`  string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`UInt32:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadUInt64()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 193 ReadInt64() - Native C++

Syntax	`ERuntimeErrorCode ReadUInt64(`
	`  WCHAR* in_Tag,`
	`  UINT64* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `UINT64* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 194 ReadUInt64() - .NET (C#)

Syntax	`UInt64 ReadUInt64(`
	`  string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`UInt64:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

	`ERuntimeErrorCode.InstanceNot-Running`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.In-dexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadFloat()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 195 ReadFloat() - Native C++

Syntax	`ERuntimeErrorCode ReadFloat( WCHAR* in_Tag, float* out_Value );`	
Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `float* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.In-dexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 196 ReadFloat() - .NET (C#)

Syntax	`float ReadFloat(`
	` string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`float:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadDouble()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 197 ReadDouble() - Native C++

Syntax	`ERuntimeErrorCode ReadDouble(`
	` WCHAR* in_Tag,`
	` double* out_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `double* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

	`ERuntimeErrorCode.In-dexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 198 ReadDouble() - .NET (C#)

Syntax	`double ReadDouble(`
	`  string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`double:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemo-ved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNot-Running`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.In-dexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadChar()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 199 ReadChar() - Native C++

Syntax	`ERuntimeErrorCode ReadChar( WCHAR* in_Tag, char* out_Value );`	
Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `char* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 200 ReadChar() - .NET (C#)

Syntax	`sbyte ReadChar( string in_Tag )`	
Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`sbyte`: Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

	`ERuntimeErrorCode.InstanceNot-Running`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.In-dexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadWChar()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 201 ReadWChar() - Native C++

Syntax	`ERuntimeErrorCode ReadWChar( WCHAR* in_Tag, WCHAR* out_Value );`	
Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll. • `WCHAR* out_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.In-dexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 202 ReadWChar() - .NET (C#)

Syntax	`char ReadWChar(`
	`  string in_Tag`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die gelesen werden soll.	
Rückgabewerte	`char:` Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte gelesen werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadSignals()

Liest mehrere Signale innerhalb eines API-Aufrufs. Wenn die Funktion das erste Mal aufgerufen wird, speichert sie in den Strukturen `SDataValueByName*` interne Informationen, um die Performanz der folgenden Aufrufe zu verbessern.

Hinweis

Um einen regelmäßigen Signalaustausch zu simulieren, erstellen Sie einmal eine Signalliste für jeden Satz an Signalen. Nutzen Sie diese Signalliste für alle weiteren Zugriffe. Erstellen Sie eine neue Liste, sobald sich der Satz an Signalen ändert.

Tabelle 7- 203 ReadSignals() - Native C++

Syntax	`ERuntimeErrorCode ReadSignals(`
	` SDataValueByName* inout_Signals,`
	` UINT32 in_SignalCount`
	`);`
	`ERuntimeErrorCode ReadSignals(`
	` SDataValueByNameWithCheck* inout_Signals,`
	` UINT32 in_SignalCount`
	` bool* out_SignalsHaveChanged`
	`);`	

Parameter	• `SDataValueByName* inout_Signals:` Enthält den Namen, den Wert und den erwarteten Typ der PLC-Variablen. Wenn der erwartete Typ `UNSPECIFIC` ist, dann wird er auf den gespeicherten Typ gesetzt, wenn die Funktion erfolgreich war. Der Typ `STRUCT` wird nicht unterstützt. • `SDataValueByNameWithCheck* inout_Signals:` Enthält den Namen, den Wert und den erwarteten Typ der PLC-Variablen. Wenn der erwartete Typ `UNSPECIFIC` ist, dann wird er auf den gespeicherten Typ gesetzt, wenn die Funktion erfolgreich war. Der Typ `STRUCT` wird nicht unterstützt. `"ValueHasChanged"` wird auf `true` gesetzt, wenn sich der Wert des Signals seit dem vorhergehenden Aufruf geändert hat. • `UINT32 in_SignalCount:` Die Anzahl der Signale, die gelesen werden. • `bool* out_SignalsHaveChanged:` Gibt `true` zurück, wenn sich der Wert von mindestens einem Signal seit dem vorhergehenden Aufruf geändert hat.	
Signalfehler	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Signaloperation ist erfolgreich.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_INDEX_OUT_OF_RANGE`	Offset oder Bits sind ungültig.

Rückgabewerte	Runtime Fehlercode	Bedingung
	SREC_OK	Die Funktion ist erfolgreich.
	SREC_INTERFACE_REMOVED	Die Instanz ist im Runtime Manager nicht registriert.
	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.
	SREC_INSTANCE_NOT_RUNNING	Der Prozess des virtuellen Controllers läuft nicht.
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.
	SREC_SIGNAL_CONFIGURATION_ERROR	Mindestens ein Signalfehler ist in der Liste.

Tabelle 7- 204 ReadSignals() - .NET (C#)

Syntax	`void ReadSignals(`
	` ref SDataValueByName[] inout_Signals`
	`)`
	`void ReadSignals(`
	` ref SDataValueByNameWithCheck[] inout_Signals`
	` out bool out_SignalsHaveChanged`
	`);`	

Parameter	• `ref SDataValueByName[] inout_Signals:` Enthält den Namen, den Wert und den erwarteten Typ der PLC-Variablen. Wenn der erwartete Typ `UNSPECIFIC` ist, dann wird er auf den gespeicherten Typ gesetzt, wenn die Funktion erfolgreich war. Der Typ `STRUCT` wird nicht unterstützt. • `ref SDataValueByNameWithCheck[] inout_Signals:` Enthält den Namen, den Wert und den erwarteten Typ der PLC-Variablen. Wenn der erwartete Typ `UNSPECIFIC` ist, dann wird er auf den gespeicherten Typ gesetzt, wenn die Funktion erfolgreich war. Der Typ `STRUCT` wird nicht unterstützt. `"ValueHasChanged"` wird auf `true` gesetzt, wenn sich der Wert des Signals seit dem vorhergehenden Aufruf geändert hat. • `out bool out_SignalsHaveChanged:` Gibt `true` zurück, wenn sich der Wert von mindestens einem Signal seit dem vorhergehenden Aufruf geändert hat.	
Rückgabewerte	`SDataValue:` Enthält den Wert und den Typ der PLC-Variablen.	
Signalfehler	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.Ok`	Die Signaloperation ist erfolgreich.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.IndexOutOfRange`	Offset oder Bits sind ungültig.

Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

7.6.5.5 I/O-Zugriff über Variablenname - Schreiben

Einzelzugriffe auf IO-Daten sind dazu geeignet, Werte, die nicht regelmäßig aktualisiert werden, auf einer grafischen Oberfläche (GUI) anzuzeigen und zu schreiben.

Hinweis

Um einen regelmäßigen Signalaustausch zu simulieren, erstellen Sie einmal eine Signalliste für jeden Satz an Signalen. Nutzen Sie diese Signalliste für alle weiteren Zugriffe. Erstellen Sie eine neue Liste, sobald sich der Satz an Signalen ändert.

Verwenden Sie für die Signallisten die Funktionen `ReadSignals()` und `WriteSignals()`.

Write()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 205 Write() - Native C++

Syntax	`ERuntimeErrorCode Write(`
	` WCHAR* in_Tag,`
	` SDataValue* in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `SDataValue* in_Value:` Enthält den Wert und den erwarteten Typ der PLC-Variablen. Die Typen `UNSPECIFIC` und `STRUCT` werden nicht unterstützt. Strukturen und Felder können durch Signallisten nachgebildet und dann über `ReadSignals()` gelesen und über `WriteSignals()` geschrieben werden.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

	SREC_INDEX_OUT_OF_RANGE	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.
	SREC_WRONG_ARGUMENT	Der erwartete Typ ist UNSPECIFIC.

Tabelle 7- 206 Write() - .NET (C#)

Syntax	`void Write(`
	`  string in_Tag`
	`  SDataValue in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `SDataValue in_Value:` Enthält den Wert und den erwarteten Typ der PLC-Variablen. Die Typen UNSPECIFIC und STRUCT werden nicht unterstützt. Strukturen und Felder können durch Signallisten nachgebildet und dann über `WriteSignals()` geschrieben werden.	
Rückgabewerte	Keine	
Ausnahmen	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime Fehlercode	Bedingung
	ERuntimeErrorCode.InterfaceRemoved	Die Instanz ist im Runtime Manager nicht registriert.
	ERuntimeErrorCode.Timeout	Die Funktion kehrt nicht rechtzeitig wieder.
	ERuntimeErrorCode.InstanceNotRunning	Der Prozess des virtuellen Controllers läuft nicht.
	ERuntimeErrorCode.IndexOutOfRange	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	ERuntimeErrorCode.DoesNotExist	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	ERuntimeErrorCode.NotSupported	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	ERuntimeErrorCode.TypeMismatch	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	ERuntimeErrorCode.NotUpToData	Die gespeicherte Variablentabelle muss aktualisiert werden.
	ERuntimeErrorCode.WrongArgument	Der erwartete Typ ist UNSPECIFIC.

WriteBool()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 207 WriteBool() - Native C++

Syntax	`ERuntimeErrorCode WriteBool(`
	` WCHAR* in_Tag,`
	` bool in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `bool in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 208 WriteBool() - .NET (C#)

Syntax	`void WriteBool(`
	` string in_Tag`
	` bool in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `bool in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.

	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMismatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 382).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteInt8()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 209 WriteInt8() - Native C++

Syntax	`ERuntimeErrorCode WriteInt8(`
	`  WCHAR* in_Tag,`
	`  INT8 in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `INT8 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 210 WriteInt8() - .NET (C#)

Syntax	`void WriteInt8(`
	` string in_Tag`
	` Int8 in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `Int8 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteInt16()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 211 WriteInt16() - Native C++

Syntax	`ERuntimeErrorCode WriteInt16(`
	` WCHAR* in_Tag,`
	` INT16 in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `INT16 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.

	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.
	SREC_INSTANCE_NOT_RUNNING	Der Prozess des virtuellen Controllers läuft nicht.
	SREC_INDEX_OUT_OF_RANGE	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 212 WriteInt16() - .NET (C#)

Syntax	`void WriteInt16(`
	`  string in_Tag`
	`  Int16 in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `Int16 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	ERuntimeErrorCode.InterfaceRemoved	Die Instanz ist im Runtime Manager nicht registriert.
	ERuntimeErrorCode.Timeout	Die Funktion kehrt nicht rechtzeitig wieder.
	ERuntimeErrorCode.InstanceNotRunning	Der Prozess des virtuellen Controllers läuft nicht.
	ERuntimeErrorCode.IndexOutOfRange	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	ERuntimeErrorCode.DoesNotExist	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	ERuntimeErrorCode.NotSupported	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	ERuntimeErrorCode.TypeMissmatch	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	ERuntimeErrorCode.NotUpToData	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteInt32()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 213 WriteInt32() - Native C++

Syntax	`ERuntimeErrorCode WriteInt32(`
	`  WCHAR* in_Tag,`
	`  INT32 in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `INT32 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 214 WriteInt32() - .NET (C#)

Syntax	`void WriteInt32(`
	`  string in_Tag`
	`  Int32 in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `Int32 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.

	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteInt64()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 215 WriteInt64() - Native C++

Syntax	`ERuntimeErrorCode WriteInt64(`
	`  WCHAR* in_Tag,`
	`  INT64 in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `INT64 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 216 WriteInt64() - .NET (C#)

Syntax	`void WriteInt64(`
	` string in_Tag`
	` Int64 in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `Int64 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteUInt8()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 217 WriteUInt8() - Native C++

Syntax	`ERuntimeErrorCode WriteUInt8(`
	` WCHAR* in_Tag,`
	` UINT8 in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `UINT8 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.

	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.
	SREC_INSTANCE_NOT_RUNNING	Der Prozess des virtuellen Controllers läuft nicht.
	SREC_INDEX_OUT_OF_RANGE	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 218 WriteUInt8() - .NET (C#)

Syntax	`void WriteUInt8(`
	`  string in_Tag`
	`  UInt8 in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `UInt8 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	ERuntimeErrorCode.InterfaceRemoved	Die Instanz ist im Runtime Manager nicht registriert.
	ERuntimeErrorCode.Timeout	Die Funktion kehrt nicht rechtzeitig wieder.
	ERuntimeErrorCode.InstanceNotRunning	Der Prozess des virtuellen Controllers läuft nicht.
	ERuntimeErrorCode.IndexOutOfRange	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	ERuntimeErrorCode.DoesNotExist	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	ERuntimeErrorCode.NotSupported	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	ERuntimeErrorCode.TypeMissmatch	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	ERuntimeErrorCode.NotUpToData	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteUInt16()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 219 WriteUInt16() - Native C++

Syntax	`ERuntimeErrorCode WriteUInt16(`
	`  WCHAR* in_Tag,`
	`  UINT16 in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `UINT16 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 220 WriteUInt16() - .NET (C#)

Syntax	`void WriteUInt16(`
	`  string in_Tag`
	`  UInt16 in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `UInt16 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.

	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteUInt32()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 221 WriteUInt32() - Native C++

Syntax	`ERuntimeErrorCode WriteUInt32(`
	`  WCHAR* in_Tag,`
	`  UINT32 in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `UINT32 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 222 WriteUInt32() - .NET (C#)

Syntax	`void WriteUInt32(`
	` string in_Tag`
	` UInt32 in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `UInt32 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteUInt64()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 223 WriteUInt64() - Native C++

Syntax	`ERuntimeErrorCode WriteUInt64(`
	` WCHAR* in_Tag,`
	` UINT64 in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `UINT64 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.

	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.
	SREC_INSTANCE_NOT_RUNNING	Der Prozess des virtuellen Controllers läuft nicht.
	SREC_INDEX_OUT_OF_RANGE	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 224 WriteUInt64() - .NET (C#)

Syntax	`void WriteUInt64(`
	` string in_Tag`
	` UInt64 in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `UInt64 in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteFloat()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 225 WriteFloat() - Native C++

Syntax	`ERuntimeErrorCode WriteFloat(`
	`  WCHAR* in_Tag,`
	`  float in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `float in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 226 WriteFloat() - .NET (C#)

Syntax	`void WriteFloat(`
	`  string in_Tag`
	`  float in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `float in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.

	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteDouble()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 227 WriteDouble() - Native C++

Syntax	`ERuntimeErrorCode WriteDouble(`
	` WCHAR* in_Tag,`
	` double in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `double in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 228 WriteDouble() - .NET (C#)

Syntax	`void WriteDouble(`
	` string in_Tag`
	` double in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `double in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteChar()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 229 WriteChar() - Native C++

Syntax	`ERuntimeErrorCode WriteChar(`
	` WCHAR* in_Tag,`
	` char in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `char in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.

	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.
	SREC_INSTANCE_NOT_RUNNING	Der Prozess des virtuellen Controllers läuft nicht.
	SREC_INDEX_OUT_OF_RANGE	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 230 WriteChar() - .NET (C#)

Syntax	`void WriteChar(`
	`  string in_Tag`
	`  sbyte in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `sbyte in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteWChar()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 231 WriteWChar() - Native C++

Syntax	`ERuntimeErrorCode WriteWChar(`
	` WCHAR* in_Tag,`
	` WCHAR in_Value`
	`);`	

Parameter	• `WCHAR* in_Tag:` Der Name der PLC-Variablen, die geschieben werden soll. • `WCHAR in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`SREC_DOES_NOT_EXIST`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`SREC_NOT_SUPPORTED`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`SREC_TYPE_MISMATCH`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 232 WriteWChar() - .NET (C#)

Syntax	`void WriteWChar(`
	` string in_Tag`
	` char in_Value`
	`)`	

Parameter	• `string in_Tag:` Der Name der PLC-Variablen, die geschrieben werden soll. • `char in_Value:` Enthält den Wert der PLC-Variablen.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.

	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Der Offset liegt außerhalb der Area-Größe. Kein Wert konnte geschrieben werden.
	`ERuntimeErrorCode.DoesNotExist`	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	`ERuntimeErrorCode.NotSupported`	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	`ERuntimeErrorCode.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteSignals()

Schreibt mehrere Signale innerhalb eines API-Aufrufs. Wenn die Funktion das erste Mal aufgerufen wird, speichert sie in den Strukturen `SDataValueByName*` interne Informationen, um die Performanz der folgenden Aufrufe zu verbessern.

Hinweis

Um einen regelmäßigen Signalaustausch zu simulieren, erstellen Sie einmal eine Signalliste für jeden Satz an Signalen. Nutzen Sie diese Signalliste für alle weiteren Zugriffe. Erstellen Sie eine neue Liste, sobald sich der Satz an Signalen ändert.

Tabelle 7- 233 WriteSignals() - Native C++

Syntax	`ERuntimeErrorCode WriteSignals(`
	`  SDataValueByName* inout_Signals,`
	`  UINT32 in_SignalCount`
	`);`	

Parameter	• `SDataValueByName* inout_Signals:` Enthält den Namen, den Wert und den erwarteten Typ der PLC-Variablen. Die Typen `UNSPECIFIC` und `STRUCT` werden nicht unterstützt. • `UINT32 in_SignalCount:` Anzahl der Signale.	
Signalfehler	Fehlercode	Bedingung
	`SREC_INDEX_OUT_OF_RANGE`	Offset oder Bits sind ungültig.
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

	SREC_INSTANCE_NOT_RUNNING	Der Prozess des virtuellen Controllers läuft nicht.
	SREC_DOES_NOT_EXIST	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	SREC_NOT_SUPPORTED	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	SREC_TYPE_MISMATCH	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 380).
	SREC_NOT_UP_TO_DATE	Die gespeicherte Variablentabelle muss aktualisiert werden.
	SREC_WRONG_ARGUMENT	Der erwartete Typ ist UNSPECIFIC.

Tabelle 7- 234 WriteSignals() - .NET (C#)

Syntax	`void WriteSignals(`
	`  SDataValueByName[] in_Signals`
	`)`	

Parameter	• SDataValueByName: Enthält den Namen, den Wert und den erwarteten Typ der PLC-Variablen. Die Typen UNSPECIFIC und STRUCT werden nicht unterstützt.	
Rückgabewerte	Keine	
Signalfehler	Fehlercode	Bedingung
	ERuntimeErrorCode.IndexOutOfRange	Offset oder Bits sind ungültig.
Ausnahmen	Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException	
	Runtime Fehlercode	Bedingung
	ERuntimeErrorCode.InterfaceRemoved	Die Instanz ist im Runtime Manager nicht registriert.
	ERuntimeErrorCode.Timeout	Die Funktion kehrt nicht rechtzeitig wieder.
	ERuntimeErrorCode.InstanceNotRunning	Der Prozess des virtuellen Controllers läuft nicht.
	ERuntimeErrorCode.DoesNotExist	Der Eintrag existiert nicht in der gespeicherten Variablentabelle.
	ERuntimeErrorCode.NotSupported	Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.
	ERuntimeErrorCode.TypeMismatch	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 382).
	ERuntimeErrorCode.NotUpToData	Die gespeicherte Variablentabelle muss aktualisiert werden.
	ERuntimeErrorCode.WrongArgument	Der erwartete Typ ist UNSPECIFIC.

7.6.6 Einstellungen für die virtuelle Zeit

GetSystemTime()

Liefert die virtuelle Systemzeit des virtuellen Controllers zurück. Liefert eine leere Struktur zurück, wenn die Funktion fehlschlägt.

Tabelle 7- 235 GetSystemTime() - Native C++

Syntax	`SYSTEMTIME GetSystemTime();`
Parameter	Keine
Rückgabewerte	`SYSTEMTIME`: Systemzeit des virtuellen Controllers.

SetSystemTime()

Setzt die virtuelle Systemzeit des virtuellen Controllers. Gültig ist eine Systemzeit zwischen "Jan 1 1970 00:00:00:000" und "Dec 31 2200 23:59:59:999".

Tabelle 7- 236 SetSystemTime() - Native C++

Syntax	`ERuntimeErrorCode SetSystemTime(`
	`  SYSTEMTIME in_SystemTime`
	`);`	

Parameter	• `SYSTEMTIME in_SystemTime:` Systemzeit, die für den virtuellen Controller gesetzt werden soll.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Wert befindet sich außerhalb der Grenzen.

SystemTime { get; set; }

Setzt oder liefert die virtuelle Systemzeit des virtuellen Controllers. Gültig ist eine Systemzeit zwischen "Jan 1 1970 00:00:00:000" und "Dec 31 2200 23:59:59:999".

Tabelle 7- 237 SystemTime { get; set; } - .NET (C#)

Syntax	`DateTime SystemTime { get; set; }`	
Parameter	Keine	
Rückgabewerte	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Wert befindet sich außerhalb der Grenzen.

GetScaleFactor()

Liefert den Skalierfaktor zurück, mit dem die virtuelle Zeit fortschreitet.

Tabelle 7- 238 GetScaleFactor() - Native C++

Syntax	`double GetScaleFactor();`
Parameter	Keine
Rückgabewerte	`double:` Skalierfaktor der virtuellen Zeit.

SetScaleFactor()

Setzt den Skalierfaktor, mit dem die virtuelle Zeit fortschreitet.

Beginnen Sie mit einem kleinen Skalierfaktor und tasten Sie sich schrittweise an einen Skalierfaktor heran, bei dem der virtuelle Controller in RUN bleibt.

Gültig ist ein Wert zwischen 0,01 und 100. Die Voreinstellung ist 1.

- Wenn der Wert kleiner 1 ist, läuft die virtuelle Zeit des virtuellen Controllers X-mal langsamer als die reale Zeit.
- Wenn der Wert größer 1 ist, läuft die virtuelle Zeit des virtuellen Controllers X-mal schneller als die reale Zeit.

Eine Änderung des Werts zur Laufzeit wird erst am Zykluskontrollpunkt wirksam.

Tabelle 7- 239 SetScaleFactor() - Native C++

Syntax	`ERuntimeErrorCode SetScaleFactor (`
	` double in_Value`
	`);`	

Parameter	• `double in_Value:` Skalierfaktor der virtuellen Zeit.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Wert befindet sich außerhalb der Grenzen.

ScaleFactor { get; set; }

Setzt oder liefert den Skalierfaktor zurück, mit dem die virtuelle Zeit fortschreitet.

Beginnen Sie mit einem kleinen Skalierfaktor und tasten Sie sich schrittweise an einen Skalierfaktor heran, bei dem der virtuelle Controller in RUN bleibt.

Gültig ist ein Wert zwischen 0,01 und 100. Die Voreinstellung ist 1.

- Wenn der Wert kleiner 1 ist, läuft die virtuelle Zeit des virtuellen Controllers X-mal langsamer als die reale Zeit.
- Wenn der Wert größer 1 ist, läuft die virtuelle Zeit des virtuellen Controllers X-mal schneller als die reale Zeit.

Eine Änderung des Werts zur Laufzeit wird erst am Zykluskontrollpunkt wirksam.

Tabelle 7- 240 ScaleFactor { get; set; } - .NET (C#)

Syntax	`double ScaleFactor { get; set; }`	
Parameter	Keine	
Rückgabewerte	`double`: Skalierfaktor der virtuellen Zeit.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Wert befindet sich außerhalb der Grenzen.

7.6.7 Zykluskontrolle

GetOperatingMode()

Liefert die Betriebsart (Seite 374) des virtuellen Controllers zurück.

Tabelle 7- 241 GetOperatingMode() - Native C++

Syntax	`EOperatingMode GetOperatingMode();`
Parameter	Keine
Rückgabewerte	`EOperatingMode`: Betriebsart des virtuellen Controllers

SetOperatingMode()

Setzt die Betriebsart des virtuellen Controllers.

Eine Änderung des Werts zur Laufzeit wird erst am Synchronisationspunkt wirksam.

Tabelle 7- 242 SetOperatingMode() - Native C++

Syntax	`void SetOperatingMode(`
	`  EOperatingMode in_OperatingMode`
	`);`	

Parameter	• `EOperatingMode in_OperatingMode:` Betriebsart des virtuellen Controllers	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

OperatingMode { get; set; }

Liefert oder setzt die Betriebsart des virtuellen Controllers.

Eine Änderung des Werts zur Laufzeit wird erst am Synchronisationspunkt wirksam.

Tabelle 7- 243 OperatingMode { get; set; } - .NET (C#)

Syntax	`EOperatingMode OperatingMode { get; set; }`	
Parameter	Keine	
Rückgabewerte	`EOperatingMode:` Betriebsart des virtuellen Controllers	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

SetSendSyncEventInDefaultModeEnabled()

Setzt den Modus `SendSyncEventInDefault`. In diesem Modus wird in der Betriebsart Default nach jedem Zyklusende das Ereignis `OnSyncPointReached` ausgelöst. Siehe OnSyncPointReached (Seite 288).

Tabelle 7- 244 SetSendSyncEventInDefaultModeEnabled() - Native C++

Syntax	`ERuntimeErrorCode SetSendSyncEventInDefaultModeEnabled( bool in_Enable );`	
Parameter	• `bool in_Enable:` Wenn `true`, dann wird in der Betriebsart Default nach jedem Zyklus das Ereignis `OnSyncPointReached` ausgelöst.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

IsSendSyncEventInDefaultModeEnabled()

Liefert den Modus `SendSyncEventInDefaultMode` zurück. Wenn die Funktion fehlschlägt, ist der Rückgabewert `false`.

Tabelle 7- 245 IsSendSyncEventInDefaultModeEnabled() - Native C++

Syntax	`bool IsSendSyncEventInDefaultModeEnabled();`
Parameter	Keine
Rückgabewerte	• `false`: Das Ereignis wird nicht ausgelöst (außer der Modus Sync-Freeze ist aktiv). • `true`: Das Ereignis wird nach jedem Zyklus ausgelöst.

IsSendSyncEventInDefaultModeEnabled { get; set; }

Liefert oder setzt den Modus `SendSyncEventInDefaultMode`. In diesem Modus wird für jede Betriebsart nach jedem Zyklusende das Ereignis `OnSyncPointReached` ausgelöst. Wenn das Ereignis auch in der Betriebsart Default empfangen werden soll, setzen Sie den Rückgabewert auf `true`. Siehe OnSyncPointReached (Seite 288).

Tabelle 7- 246 IsSendSyncEventInDefaultModeEnabled { get; set; } - .NET (C#)

Syntax	`bool IsSendSyncEventInDefaultModeEnabled { get; set;}`	
Parameter	Keine	
Rückgabewerte	• `false`: Das Ereignis wird nicht ausgelöst (außer der Modus Sync-Freeze ist aktiv). • `true`: Das Ereignis wird nach jedem Zyklus ausgelöst.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

GetOverwrittenMinimalCycleTime_ns()

Liefert die überschriebene minimale Zykluszeit (in Nanosekunden), die in den Betriebsarten `SingleStep_CT` und `SingleStep_CPT` verwendet wird.

Tabelle 7- 247 GetOverwrittenMinimalCycleTime_ns() - Native C++

Syntax	`INT64 GetOverwrittenMinimalCycleTime_ns();`
Parameter	Keine
Rückgabewerte	`INT64`: Die überschriebene minimale Zykluszeit in Nanosekunden.

SetOverwrittenMinimalCycleTime_ns()

Setzt die überschriebene minimale Zykluszeit (in Nanosekunden), die in den Betriebsarten `SingleStep_CT` und `SingleStep_CPT` verwendet wird.

Gültig ist ein Wert zwischen 0 und 6000000000. Die Voreinstellung sind 100 ms.

Eine Änderung des Werts zur Laufzeit wird erst am Zykluskontrollpunkt wirksam.

Tabelle 7- 248 SetOverwrittenMinimalCycleTime_ns() - Native C++

Syntax	`ERuntimeErrorCode SetOverwrittenMinimalCycleTime_ns( INT64 in_CycleTime_ns );`	
Parameter	• `INT64 in_CycleTime_ns:` Die überschriebene minimale Zykluszeit in Nanosekunden.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Wert befindet sich außerhalb der Grenzen.

OverwrittenMinimalCycleTime_ns { get; set; }

Liefert oder setzt die überschriebene minimale Zykluszeit in Nanosekunden, die in den Betriebsarten `SingleStep_CT` und `SingleStep_CPT` verwendet wird.

Gültig ist ein Wert zwischen 0 und 6000000000. Die Voreinstellung sind 100 ms.

Eine Änderung des Werts zur Laufzeit wird erst am Zykluskontrollpunkt wirksam.

Tabelle 7- 249 OverwrittenMinimalCycleTime_ns { get; set; } - .NET (C#)

Syntax	`Int64 OverwrittenMinimalCycleTime_ns { get; set; }`	
Parameter	Keine	
Rückgabewerte	`Int64:` Die überschriebene minimale Zykluszeit in Nanosekunden.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Wert befindet sich außerhalb der Grenzen.

RunToNextSyncPoint()

Wenn der virtuelle Controller in einer SingleStep-Betriebsart läuft, wird er am Synchronisationspunkt angehalten (Freeze-Zustand). Die Funktion `RunToNextSyncPoint()` hebt den Freeze-Zustand auf. Der virtuelle Controller läuft bis zum nächsten Synchronisationspunkt weiter.

Tabelle 7- 250 RunToNextSyncPoint() - Native C++

Syntax	`ERuntimeErrorCode RunToNextSyncPoint();`	
Parameter	Keine	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.

Tabelle 7- 251 RunToNextSyncPoint() - .NET (C#)

Syntax	`void RunToNextSyncPoint();`	
Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

StartProcessing()

Wenn der virtuelle Controller in einer TimespanSynchronized-Betriebsart läuft, wird er am Synchronisationspunkt angehalten (Freeze-Zustand). Die Funktion `StartProcessing()` hebt den Freeze-Zustand auf. Der virtuelle Controller läuft nun mindestens so lange wie angefordert, bevor er beim nächsten Synchronisationspunkt wieder in den Freeze-Zustand wechselt.

Tabelle 7- 252 StartProcessing() - Native C++

Syntax	`ERuntimeErrorCode StartProcessing(`
	`  INT64 in_MinimalTimeToRun_ns`
	`);`	

Parameter	• `INT64 in_MinimalTimeToRun_ns:` Die minimale virtuelle Zeit (in Nanosekunden), die der virtuelle Controller läuft, bevor er in den Freeze-Zustand wechselt.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht.
	`SREC_WRONG_ARGUMENT`	Der Wert ist kleiner als 0.

Tabelle 7- 253 StartProcessing() - .NET (C#)

Syntax	`void StartProcessing(`
	`  Int64 in_MinimalTimeToRun_ns`
	`);`	

Parameter	• `Int64 in_MinimalTimeToRun_ns:` Die minimale virtuelle Zeit (in Nanosekunden), die der virtuelle Controller läuft, bevor er in den Freeze-Zustand wechselt.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.WrongArgument`	Der Wert ist kleiner als 0.

Weitere Informationen

Weitere Informationen siehe Kapitel Virtuelles Zeitverhalten (Seite 88), Simulation anhalten (Seite 91).

SetCycleTimeMonitoringMode()

Mit dieser Funktion kann die Quelle für den Timer zur maximalen Zykluszeitüberwachung geändert werden.

Tabelle 7- 254 SetCycleTimeMonitoringMode() - Native C++

Syntax	`ERuntimeErrorCode SetCycleTimeMonitoringMode(`
	`  ECycleTimeMonitoringMode in_CycleTimeMonitoringMode`
	`)`
	`ERuntimeErrorCode SetCycleTimeMonitoringMode(`
	`  ECycleTimeMonitoringMode in_CycleTimeMonitoringMode,`
	`  INT64 in_MaxCycleTime_ns`
	`)`	

Parameter	• `ECycleTimeMonitoringMode in_CycleTimeMonitoringMode:` Wählen Sie eine der folgenden Optionen für die maximale Zykluszeitüberwachung: – `SRCTMM_DOWNLOADED:` Als maximale Zykluszeitüberwachung wird die maximale Zykluszeit aus dem Projekt verwendet, das aus STEP 7 heruntergeladen wurde. – `SRCTMM_IGNORED` **(Voreinstellung)**: Als maximale Zykluszeitüberwachung wird ein Timer-Wert von einer Minute verwendet, um einen möglichen Fehler beim Überlauf zyklischer Ereignisse zu verhindern. Siehe Überwachung Überlauf (Seite 404). – `SRCTMM_SPECIFIED:` Als maximale Zykluszeitüberwachung wird ein Wert verwendet, der über den Parameter `in_MaxCycleTime_ns` spezifiziert werden kann. Voreinstellung: 150 ms. • `INT64 in_MaxCycleTime_ns:` Der anwenderspezifische Wert für die maximale Zykluszeitüberwachung. Gültig ist ein Wert zwischen 1000000 und 60000000000 ns (1 Millisekunde bis 1 Minute). Wenn in der API kein Wert spezifiziert wird, gilt die Voreinstellung von 150 ms.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Modus für die Zykluszeitüberwachung ist ungültig.
	`SREC_INDEX_OUT_OF_RANGE`	Der anwenderspezifische Wert für die maximale Zykluszeitüberwachung befindet sich außerhalb der Grenzen.

Tabelle 7- 255 SetCycleTimeMonitoringMode() - .NET (C#)

Syntax	`void SetCycleTimeMonitoringMode(`
	` ECycleTimeMonitoringMode in_CycleTimeMonitoringMode`
	`)`
	
	`void SetCycleTimeMonitoringMode(`
	` ECycleTimeMonitoringMode in_CycleTimeMonitoringMode,`
	` Int64 in_MaxCycleTime_ns`
	`)`	

Parameter	• `ECycleTimeMonitoringMode in_CycleTimeMonitoringMode:` Wählen Sie eine der folgenden Optionen für die maximale Zykluszeitüberwachung: – `ECycleTimeMonitoringMode.Downloaded:` Als maximale Zykluszeitüberwachung wird die maximale Zykluszeit aus dem Projekt verwendet, das aus STEP 7 heruntergeladen wurde. – `ECycleTimeMonitoringMode.Ignored` **(Voreinstellung)**: Als maximale Zykluszeitüberwachung wird ein Timer-Wert von einer Minute verwendet, um einen möglichen Fehler beim Überlauf zyklischer Ereignisse zu verhindern. Siehe Überwachung Überlauf (Seite 404). – `ECycleTimeMonitoringMode.Specified:` Als maximale Zykluszeitüberwachung wird ein Wert verwendet, der über den Parameter `in_MaxCycleTime_ns` spezifiziert werden kann. Voreinstellung: 150 ms. • `Int64 in_MaxCycleTime_ns:` Der anwenderspezifische Wert für die maximale Zykluszeitüberwachung. Gültig ist ein Wert zwischen 1000000 und 60000000000 ns (1 Millisekunde bis 1 Minute). Wenn in der API kein Wert spezifiziert wird, gilt die Voreinstellung von 150 ms.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Modus für die Zykluszeitüberwachung ist ungültig.
	`ERuntimeErrorCode.IndexOutOfRange`	Der anwenderspezifische Wert für die maximale Zykluszeitüberwachung befindet sich außerhalb der Grenzen.

GetCycleTimeMonitoringMode()

Diese Funktion liefert eine Information über die Quelle für den Timer zur maximalen Zykluszeitüberwachung zurück.

Tabelle 7- 256 GetCycleTimeMonitoringMode() - Native C++

Syntax	`ERuntimeErrorCode GetCycleTimeMonitoringMode(`
	`  ECycleTimeMonitoringMode* out_CycleTimeMonitoringMode,`
	`  INT64* out_MaxCycleTime_ns`
	`)`	

Parameter	• `ECycleTimeMonitoringMode* out_CycleTimeMonitoringMode:` Der konfigurierte Modus für die Zykluszeitüberwachung. Die Voreinstellung ist `SRCTM_IGNORED`. • `INT64 in_MaxCycleTime_ns:` Der anwenderspezifische Wert für die maximale Zykluszeitüberwachung. Wenn in der API kein Wert spezifiziert ist, wird der voreingestellte Wert 150 ms zurückgegeben.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 257 GetCycleTimeMonitoringMode() - .NET (C#)

Syntax	`void GetCycleTimeMonitoringMode(`
	`  out ECycleTimeMonitoringMode out_CycleTimeMonitoringMode,`
	`  out Int64 out_MaxCycleTime_ns`
	`)`	

Parameter	• `ECycleTimeMonitoringMode out_CycleTimeMonitoringMode:` Der konfigurierte Modus für die Zykluszeitüberwachung. Die Voreinstellung ist `ECycleTimeMonitoringMode.Ignored`. • `Int64 in_MaxCycleTime_ns:` Der anwenderspezifische Wert für die maximale Zykluszeitüberwachung. Wenn in der API kein Wert spezifiziert ist, wird der voreingestellte Wert 150 ms zurückgegeben.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.8 Azyklische Dienste

7.6.8.1 Übersicht

Die azyklischen Dienste umfassen bei PLCSIM Advanced:

- Lese- und Schreibvorgänge von Parameter- und Status-Daten vom Anwenderprogramm der SPS in die Peripheriemodule
- Alarm- und Ereignismeldungen, die die Peripheriemodule an die CPU senden.

Lese- und Schreibvorgänge

Vom Anwenderprogramm (TIA Portal) getriggerte Ereignisse, die sich für die Notifizierung angemeldet haben:

Tabelle 7- 258 Ereignisse: Lese- und Schreibvorgänge

SFB	Name	API-Methode (Meldung)	API-Ereignis zum Triggern des SFB
52	RDREC	ReadRecordDone (Seite 260)	OnDataRecordRead (Seite 292)
53	WRREC	WriteRecordDone (Seite 260)	OnDataRecordWrite (Seite 292)

Ablaufschema

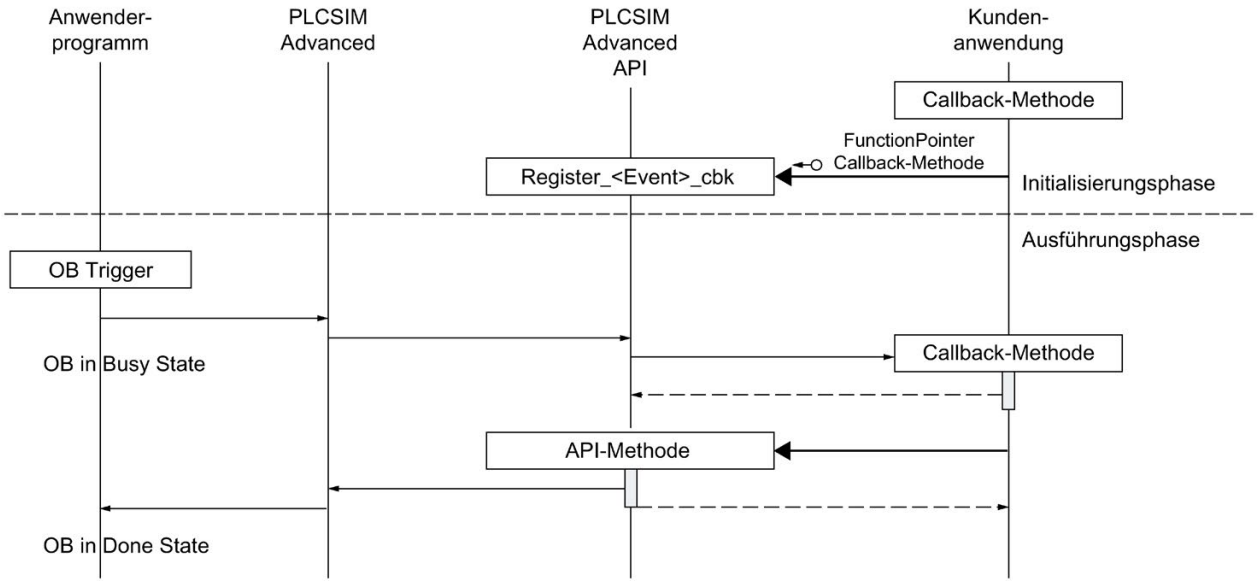


Bild 7-4 Ablaufschema Lese- und Schreibvorgänge

API-Methoden und zugehörige Ereignisse

Ereignisse, die von Peripheriemodulen ausgelöst werden, und zugehörige API-Methoden:

Tabelle 7- 259 API-Methoden und zugehörige Ereignisse

OB	Name	API-Methode zum Triggern des OB (Anfrage)	API-Ereignis nach OB-Ausführung (Meldung)
82	Diagnostic error Interrupt	AlarmNotification (Seite 263)	OnAlarmNotificationDone (Seite 294)
4x	Hardware Interrupt	ProcessEvent (Seite 267)	OnProcessEventDone (Seite 295)
83	Pull or Plug of module	PullOrPlugEvent (Seite 269)	OnPullOrPlugEventDone (Seite 296)
55	Status	StatusEvent (Seite 271)	OnStatusEventDone (Seite 297)
57	Profile	ProfileEvent (Seite 272)	OnProfileEventDone (Seite 298)
56	Update	UpdateEvent (Seite 273)	OnUpdateEventDone (Seite 299)
86	Rack or station failure	RackOrStationFaultEvent (Seite 276)	OnRackOrStationFaultEvent-Done (Seite 300)

Ablaufschema

Ablaufschema bei der Simulation von Ereignissen, die von Peripheriemodulen ausgelöst werden.

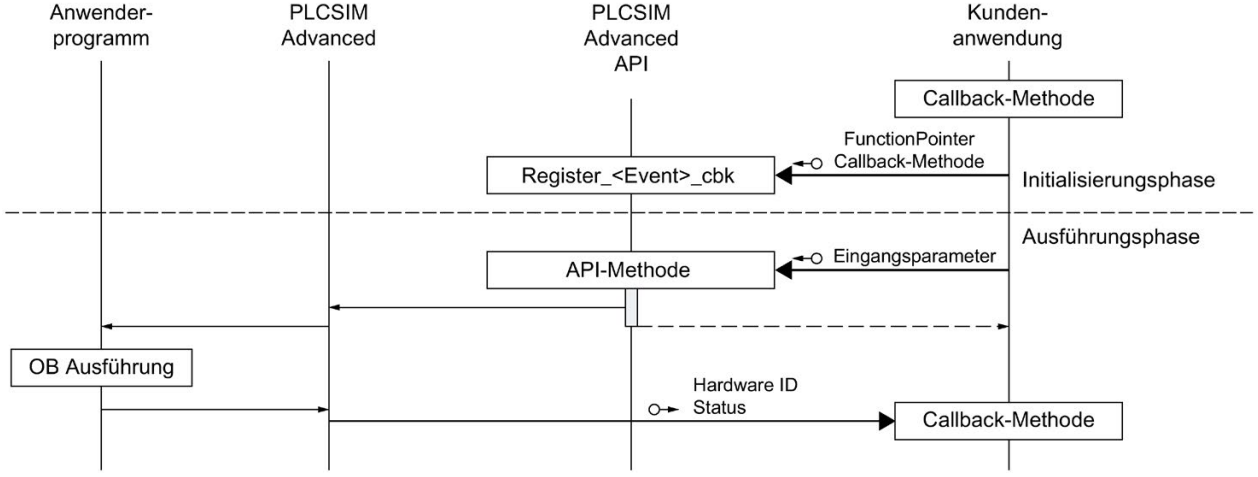


Bild 7-5 Ablaufschema bei der Simulation von Ereignissen

7.6.8.2 ReadRecordDone / WriteRecordDone

ReadRecordDone()

Mit dieser API-Methode meldet die Simulation eines IO-Moduls an die CPU, dass das asynchrone Lesen eines Datensatzes abgeschlossen ist. Die Simulation stellt dabei die gelesenen Informationen zur Verfügung.

Tabelle 7- 260 ReadRecordDone() - Native C++

Syntax	`ERuntimeErrorCode ReadRecordDone(`
	` SDataRecordInfo in_RecordInfo,`
	` BYTE* in_Data,`
	` UINT32 in_Status`
	`);`	

Parameter	• `SDataRecordInfo in_RecordInfo:` Struktur, die die Datensatz-Information enthält. Siehe SDataRecordInfo (Seite 362). • `BYTE* in_Data:` Byte-Array des gelesenen Datensatzes mit der Länge, die definiert ist über `DataSize` in der Struktur `SDataRecordInfo`. • `UINT32 in_Status:` Status der Auftragsausführung	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_DOES_NOT_EXIST`	Die HW-Kennung des Moduls existiert nicht.
	`SREC_INDEX_OUT_OF_RANGE`	Das Byte-Array des gelesenen Datensatzes überschreitet die Länge `DDATARECORD_MAX_SIZE` = 64000.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 261 ReadRecordDone() - .NET (C#)

Syntax	`void ReadRecordDone( SDataRecordInfo in_RecordInfo, BYTE[] in_Data, UInt32 in_Status );`	
Parameter	• `SDataRecordInfo in_RecordInfo:` Struktur, die die Datensatz-Information enthält. Siehe SDataRecordInfo (Seite 362). • `BYTE[] in_Data:` Byte-Array des gelesenen Datensatzes mit der Länge, die definiert ist über `DataSize` in der Struktur `SDataRecordInfo`. • `UInt32 in_Status:` Status der Auftragsausführung	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.DoesNotExist`	Die HW-Kennung des Moduls existiert nicht.
	`ERuntimeErrorCode.IndexOutOfRange`	Das Byte-Array des gelesenen Datensatzes überschreitet die Länge `DataRecordMaxSize` = 64000.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

WriteRecordDone()

Mit dieser API-Methode meldet die Simulation eines IO-Moduls an die CPU, dass das asynchrone Schreiben eines Datensatzes abgeschlossen ist.

Tabelle 7- 262 WriteRecordDone() - Native C++

Syntax	`ERuntimeErrorCode WriteRecordDone( SDataRecordInfo in_RecordInfo, UINT32 in_Status );`	
Parameter	• `SDataRecordInfo in_RecordInfo:` Struktur, die die Datensatz-Information enthält. Siehe SDataRecordInfo (Seite 362). • `UINT32 in_Status:` Status der Auftragsausführung	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_DOES_NOT_EXIST`	Die HW-Kennung des Moduls existiert nicht.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 263 WriteRecordDone() - .NET (C#)

Syntax	`void WriteRecordDone(`
	` SDataRecordInfo in_RecordInfo,`
	` UInt32 in_Status`
	`);`	

Parameter	• `SDataRecordInfo in_RecordInfo:` Struktur, die die Datensatz-Information enthält. Siehe SDataRecordInfo (Seite 362). • `UInt32 in_Status:` Status der Auftragsausführung	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.DoesNotExist`	Die HW-Kennung des Moduls existiert nicht.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.8.3 AlarmNotification

AlarmNotification()

Diese Funktion löst Diagnosemeldungen nach PROFINET-Standard aus.

Jeder Aufruf dieser Funktion ruft den OB 82 einmal auf, unabhängig von der Anzahl und vom Schweregrad der übermittelten Diagnoseeinträge.

Tabelle 7- 264 AlarmNotification() - Native C++

Syntax	`ERuntimeErrorCode AlarmNotification(`
	` UINT16 in_HardwareIdentifier,`
	` UINT16 in_ModuleState,`
	` UINT16 in_NumberOfDiagnosisEvents,`
	`SDiagExtChannelDescription* in_ArrayOfDiagnosisEvents,`
	`UINT16* out_SequenceNumber`
	`);`

Parameter	• `UINT16 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das den Diagnoseeintrag sendet. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `UINT16 in_ModuleState:` Modulzustand. Gültig sind folgende Zustände: DMODULE_STATE_OK = 0, DMODULE_STATE_ERROR = 1, DMODULE_STATE_MAINT_DEMANDED = 2, DMODULE_STATE_MAINT_REQUIRED = 4 Der Parameter `in_ModuleState` ergibt sich aus der Summe (Veroderung) der Schweregrade im Feld `SDiagExtChannelDescription`. Wenn z. B. für ein Modul sowohl bei "Wartungsanforderung" als auch bei "Wartungsbedarf" ein Diagnosealarm generiert werden soll, dann wählen Sie als Modulstatus "6". • `UINT16 in_NumberOfDiagnosisEvents:` Mehrere Diagnoseeinträge können mit einem einzigen API-Aufruf an die CPU gesendet werden. Gültiger Bereich: 0 bis 16. 0 bedeutet, dass kein Diagnoseeintrag für das Submodul oder den Kanal erscheinen soll. • `SDiagExtChannelDescription* in_ArrayOfDiagnosisEvents:` Zeiger auf ein Feld mit Diagnoseeinträgen. Das Feld muss mit der Anzahl der Diagnoseeinträge übereinstimmen. Es kann auch ein Null-Zeiger sein. Definitionen siehe SDiagExtChannelDescription (Seite 365). • `UINT16* out_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinander folgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1.

Rückgabewerte	Runtime Fehlercode	Bedingung
	SREC_OK	Die Funktion ist erfolgreich.
	SREC_INTERFACE_REMOVED	Die Instanz ist im Runtime Manager nicht registriert.
	SREC_WRONG_MODULE_STATE	Das Modul ist momentan gezogen.
	SREC_DOES_NOT_EXIST	Die HW-Kennung des Moduls existiert nicht.
	SREC_WRONG_MODULE_TYPE	Die Kanalnummer existiert für das Modul nicht.
	SREC_WRONG_ARGUMENT	Der Wert für den Modulzustand ist ungültig.
	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 265 AlarmNotification() - .NET (C#)

Syntax	`void AlarmNotification(`
	` ushort in_HardwareIdentifier,`
	` ushort in_ModuleState,`
	` ushort in_NumberOfDiagnosisEvents,`
	`SDiagExtChannelDescription [] in_ArrayOfDiagnosisEvents,`
	`Out ushort out_SequenceNumber`
	`);`

Parameter	• `ushort in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das den Diagnoseeintrag sendet. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `ushort in_ModuleState:` Modulzustand. Gültig sind folgende Zustände: ModuleState.Ok = 0, ModuleState.Error = 1, ModuleState.MaintenanceDemanded = 2, ModuleState.MaintenanceRequired = 4 Der Parameter `in_ModuleState` ergibt sich aus der Summe (Veroderung) der Schweregrade im Feld `SDiagExtChannelDescription`. Wenn z. B. für ein Modul sowohl bei "Wartungsanforderung" als auch bei "Wartungsbedarf" ein Diagnosealarm generiert werden soll, dann wählen Sie als Modulstatus "6". • `ushort in_NumberOfDiagnosisEvents` Mehrere Diagnoseeinträge können mit einem einzigen API-Aufruf an die CPU gesendet werden. Gültiger Bereich: 0 bis 16. 0 bedeutet, dass kein Diagnoseeintrag für das Submodul oder den Kanal erscheinen soll. • `SDiagExtChannelDescription [] in_ArrayOfDiagnosisEvents:` Zeiger auf ein Feld mit Diagnoseeinträgen. Das Feld muss mit der Anzahl der Diagnoseeinträge übereinstimmen. Es kann auch ein Null-Zeiger sein. Definitionen siehe SDiagExtChannelDescription (Seite 365). • `Out ushort out_SequenceNumber:` PLCSIM Advanced weist jedem Alarm-Ereignis eine eindeutige aufeinander folgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1.
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`

Runtime Fehlercode	Bedingung
`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
`ERuntimeErrorCode.WrongModuleState`	Das Modul ist momentan gezogen.
`ERuntimeErrorCode.DoesNotExist`	Die HW-Kennung des Moduls existiert nicht.

	`ERuntimeErrorCode.WrongArgument`	Der Wert für den Modulzustand ist ungültig.
	`ERuntimeErrorCode.WrongModuleType`	Die Kanalnummer existiert für das Modul nicht.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

Beispiel	`ushort seqNumber;`
	`var In_ArrayOfDiagnosisEvent = new SDiagExtChannelDescription[] {`
	`new SDiagExtChannelDescription() {ChannelNumber = 0x8000, ErrorType = 0x0001, ExtErrorType = 0, Direction = EDiagProperty.Appear,Severity =EDiagSeverity.MaintDemanded},`
	`new SDiagExtChannelDescription() {ChannelNumber = 0x8000, ErrorType = 0x0002, ExtErrorType = 0, Direction = EDiagProperty.Appear,Severity =EDiagSeverity.Failure},`
	`new SDiagExtChannelDescription() {ChannelNumber = 0x8000, ErrorType = 0x0003, ExtErrorType = 0, Direction = EDiagProperty.Appear,Severity =EDiagSeverity.MaintRequired},`
	`Instance.AlarmNotification(269, 7, 3, In_ArrayOfDiagnosisEvent, out seqNumber);`
	`//ModuleState parameter is sum of the severities in the SDiagExtChannelDescription array above: 4+2+1`	

7.6.8.4 ProcessEvent

ProcessEvent()

Mit dieser Funktion können Prozessalarme von zentralen und dezentralen Eingangsmodulen simuliert werden.

Tabelle 7- 266 ProcessEvent() - Native C++

Syntax	`ERuntimeErrorCode ProcessEvent(`
	` UINT16 in_HardwareIdentifier,`
	` UINT16 in_Channel,`
	`EProcessEventType in_ProcessEventType,`
	`UINT16* out_SequenceNumber);`	

Parameter	• `UINT16 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Prozessereignis sendet. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `UINT16 in_Channel:` Der Kanal des IO-Moduls, das das Prozessereignis sendet. • `EProcessEventType in_ProcessEventType:` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen für S7-Module, siehe EProcessEventType (Seite 390). • `UINT16* out_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinanderfolgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_WRONG_MODULE_STATE`	Das Modul ist momentan gezogen.
	`SREC_DOES_NOT_EXIST`	Die HW-Kennung des Moduls existiert nicht.
	`SREC_NOT_SUPPORTED_BY_MODULE`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 267 ProcessEvent() - .NET (C#)

Syntax	`void ProcessEvent(`
	`ushort in_HardwareIdentifier,`
	`ushort in_Channel,`
	`EProcessEventType in_ProcessEventType,`
	`Out ushort out_SequenceNumber`
	`);`	

Parameter	• `ushort in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Prozessereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `ushort in_Channel:` Der Kanal des IO-Moduls, das das Prozessereignis generiert. • `EProcessEventType in_ProcessEventType:` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen für S7-Module, siehe EProcessEventType (Seite 390). • `Out ushort out_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinanderfolgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.WrongModuleState`	Das Modul ist momentan gezogen.
	`ERuntimeErrorCode.DoesNotExist`	Die HW-Kennung des Moduls existiert nicht.
	`ERuntimeErrorCode.NotSupportedByModule`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.8.5 PullOrPlugEvent

PullOrPlugEvent()

Diese Funktion löst Ziehen/Stecken-Ereignisse aus. Für diese Ereignisse wird der Alarm-OB (OB 83) "Pull or plug of modules" ausgeführt.

Tabelle 7- 268 PullOrPlugEvent() - Native C++

Syntax	`ERuntimeErrorCode PullOrPlugEvent(`
	`UINT16 in_HardwareIdentifier,`
	`EPullOrPlugEventType in_PullOrPlugEventType,`
	`UINT16* out_SequenceNumber`
	`);`	

Parameter	• `UINT16 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Ziehen/Stecken-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `EPullOrPlugEventType in_PullOrPlugEventType:` Ein Wert aus der Liste der vordefinierten Typen von Ziehen/Stecken-Ereignissen, siehe EPullOrPlugEventType (Seite 389). • `UINT16* out_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinander folgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_WRONG_MODULE_STATE`	Das Modul ist momentan gezogen.
	`SREC_WRONG_MODULE_TYPE`	Der falsche Modultyp wurde gewählt. Z. B. wenn ein Onboard-IO einer Kompakt-CPU gezogen werden soll.
	`SREC_NOT_SUPPORTED_BY_MODULE`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`SREC_DOES_NOT_EXIST`	Die Hardware-Kennung des Moduls existiert nicht.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 269 PullOrPlugEvent() - .NET (C#)

Syntax	`void PullOrPlugEvent(`
	`ushort in_HardwareIdentifier,`
	`EPullOrPlugEventType in_PullOrPlugEventType,`
	`Out ushort out_SequenceNumber`
	`);`	

Parameter	• `ushort in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Ziehen/Stecken-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `EPullOrPlugEventType in_PullOrPlugEventType:` Ein Wert aus der Liste der vordefinierten Typen von Ziehen/Stecken-Ereignissen, siehe EPullOrPlugEventType (Seite 389). • `Out ushort out_SequenceNumber` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinander folgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.WrongModuleState`	Das Modul ist momentan gezogen.
	`ERuntimeErrorCode.WrongModuleType`	Der falsche Modultyp wurde gewählt. Z. B. wenn ein Onboard-IO einer Kompakt-CPU gezogen werden soll.
	`ERuntimeErrorCode.DoesNotExist`	Die HW-Kennung des Moduls existiert nicht.
	`ERuntimeErrorCode.NotSupportedByModule`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.8.6 StatusEvent

StatusEvent()

Diese Funktion wird benutzt, um den Status-Ereignis-OB (OB 55) auszulösen. Status-Ereignisse werden nur für Module in einem dezentralen IO-System unterstützt.

Tabelle 7- 270 StatusEvent() - Native C++

Syntax	`ERuntimeErrorCode StatusEvent(`
	`UINT16 in_HardwareIdentifier,`
	`UINT16 in_Specifier`
	`);`	

Parameter	• `UINT16 in_HardwareIdentifier:` Die HW-Kennung des Moduls, das das Status-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `UINT16 in_Specifier:` Der Parameter wird als Alarm-Specifier an das Alarm-Telegramm übergeben. Er ist als Eingangsparameter des OB 55-Aufrufs verfügbar.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_WRONG_MODULE_STATE`	Das Modul ist momentan gezogen.
	`SREC_NOT_SUPPORTED_BY_MODULE`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`SREC_DOES_NOT_EXIST`	Die HW-Kennung des Moduls existiert nicht.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 271 StatusEvent() - .NET (C#)

Syntax	`void StatusEvent(`
	`ushort in_HardwareIdentifier,`
	`ushort in_Specifier`
	`);`	

Parameter	• `ushort in_HardwareIdentifier:` Die HW-Kennung des Moduls, das das Status-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `ushort in_Specifier:` Der Parameter wird als Alarm-Specifier an das Alarm-Telegramm übergeben. Er ist als Eingangsparameter des OB 55-Aufrufs verfügbar.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung

	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.WrongModuleState`	Das Modul ist momentan gezogen.
	`ERuntimeErrorCode.DoesNotExist`	Die HW-Kennung des Moduls existiert nicht.
	`ERuntimeErrorCode.NotSupportedByModule`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.8.7 ProfileEvent

ProfileEvent()

Diese Funktion wird benutzt, um den Profile-Ereignis OB (OB 57) auszulösen. Profile-Ereignisse werden nur für Module in einem dezentralen IO-System unterstützt.

Tabelle 7- 272 ProfileEvent() - Native C++

Syntax	`ERuntimeErrorCode ProfileEvent(`
	`UINT16 in_HardwareIdentifier,`
	`UINT16 in_Specifier`
	`);`	

Parameter	• `UINT16 in_HardwareIdentifier:` Die HW-Kennung des Moduls, das das Profile-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `UINT16 in_Specifier:` Der Parameter wird als Alarm-Specifier an das Alarm-Telegramm übergeben. Er ist als Eingangsparameter des OB 57-Aufrufs verfügbar.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_WRONG_MODULE_STATE`	Das Modul ist momentan gezogen.
	`SREC_NOT_SUPPORTED_BY_MODULE`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`SREC_DOES_NOT_EXIST`	Die HW-Kennung des Moduls existiert nicht.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 273 ProfileEvent() - .NET (C#)

Syntax	`void ProfileEvent(`
	`ushort in_HardwareIdentifier,`
	`ushort in_Specifier`
	`);`	

Parameter	• `ushort in_HardwareIdentifier:` Die HW-Kennung des Moduls, das das Profile-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `ushort in_Specifier:` Der Parameter wird als Alarm-Specifier an das Alarm-Telegramm übergeben. Er ist als Eingangsparameter des OB 57-Aufrufs verfügbar.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.WrongModuleState`	Das Modul ist momentan gezogen.
	`ERuntimeErrorCode.DoesNotExist`	Die HW-Kennung des Moduls existiert nicht.
	`ERuntimeErrorCode.NotSupportedByModule`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.8.8 UpdateEvent

UpdateEvent()

Diese Funktion wird benutzt, um den Update-Ereignis-OB (OB 56) auszulösen. Update-Ereignisse werden nur für Module in einem dezentralen IO-System unterstützt.

Tabelle 7- 274 UpdateEvent() - Native C++

Syntax	`ERuntimeErrorCode UpdateEvent(`
	`UINT16 in_HardwareIdentifier,`
	`UINT16 in_Specifier`
	`);`	

Parameter	• `UINT16 in_HardwareIdentifier:` Die HW-Kennung des Moduls, das das Update-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `UINT16 in_Specifier:` Der Parameter wird als Alarm-Specifier an das Alarm-Telegramm übergeben. Er ist als Eingangsparameter des OB 56-Aufrufs verfügbar.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.

	SREC_INTERFACE_REMOVED	Die Instanz ist im Runtime Manager nicht registriert.
	SREC_WRONG_MODULE_STATE	Das Modul ist momentan gezogen.
	SREC_NOT_SUPPORTED_BY_MODULE	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	SREC_DOES_NOT_EXIST	Die HW-Kennung des Moduls existiert nicht.
	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 275 UpdateEvent() - .NET (C#)

Syntax	`void UpdateEvent(`
	`ushort in_HardwareIdentifier,`
	`ushort in_Specifier`
	`);`	

Parameter	• `ushort in_HardwareIdentifier:` Die HW-Kennung des Moduls, das das Update-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören. • `ushort in_Specifier:` Der Parameter wird als Alarm-Specifier an das Alarm-Telegramm übergeben. Er ist als Eingangsparameter des OB 56-Aufrufs verfügbar.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.WrongModuleState`	Das Modul ist momentan gezogen.
	`ERuntimeErrorCode.DoesNotExist`	Die HW-Kennung des Moduls existiert nicht.
	`ERuntimeErrorCode.NotSupportedByModule`	Das Modul wird nicht unterstützt bei dieser Anwenderaktion.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.8.9 GetConfiguredProcessEvent

GetConfiguredProcessEvents()

Mit dieser API-Methode können zur Laufzeit die im TIA Portal konfigurierten Prozessalarme ausgelesen werden.

Wenn keine Prozessereignisse vorliegen, wird `SREC_OK` zurückgegeben. Der Wert für `EventsCount` ist dann 0.

Tabelle 7- 276 GetConfiguredProcessEvents() - Native C++

Syntax	`ERuntimeErrorCode GetConfiguredProcessEvents(`
	`UINT16* out_EventsCount,`
	`);`	

Parameter	• `SConfiguredProcessEvents* inout_ProcessEvents:` Zeiger oder Referenz auf einen benutzerdefinierten Speicher, der das Feld mit den heruntergeladenen konfigurierten Prozessereignissen enthält. Die Struktur SConfiguredProcessEvents (Seite 363) enthält Informationen über diese Prozessereignisse. • `UINT16* out_EventsCount:` Zeiger oder Referenz auf eine Variable, die die Anzahl der konfigurierten Prozessereignisse enthält.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 277 GetConfiguredProcessEvents() - .NET (C#)

Syntax	`SConfiguredProcessEvents [] GetConfiguredProcessEvents(` `);`	
Parameter	Keine	
Rückgabewerte	Feld mit konfigurierten Prozessereignissen und Feldgröße ergeben die Anzahl der konfigurierten Prozessereignisse.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.8.10 RackOrStationFaultEvent

Beschreibung

Diese Funktion wird benutzt, um den RackOrStationFault-Ereignis OB (OB 86) auszulösen. DIese Ereignisse werden nur bei dezentralen Geräten unterstützt.

Tabelle 7- 278 RackOrStationFaultEvent() - Native C++

Syntax	`ERuntimeErrorCode RackOrStationFaultEvent(`
	`  UINT16 in_HardwareIdentifier,`
	`ERackOrStationFaultType in_EventType`
	`);`	

Parameter	• `UINT16 in_HardwareIdentifier:` Die HW-Kennung des Geräts, das das Ereignis sendet. • `ERackOrStationFaultType in_EventType:` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen, siehe ERack-OrStationFaultType (Seite 392).	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Instanz ist im Runtime Manager nicht registriert.
	`SREC_WRONG_MODULE_TYPE`	Die spezifizierte HW-Kennung ist nicht die eines dezentralen Geräts.
	`SREC_WRONG_MODULE_STATE`	Das Gerät mit der spezifizierten HW-Kennung meldet bereits den Zustand Fault/Return.
	`SREC_DOES_NOT_EXIST`	Die spezifizierte HW-Kennung des Geräts existiert nicht.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 279 RackOrStationFaultEvent() - .NET (C#)

Syntax	`void RackOrStationFaultEvent(`
	`  ushort in_HardwareIdentifier,`
	
	`ERackOrStationFaultType in_EventType`
	`);`	

Parameter	• `ushort in_HardwareIdentifier:` Die HW-Kennung des Geräts, das das Ereignis sendet. • `ERackOrStationFaultType in_EventType:` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen, siehe ERack-OrStationFaultType (Seite 392).	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemo-ved`	Die Instanz ist im Runtime Manager nicht registriert.

	`ERuntimeErrorCode.DoesNotExist`	Die spezifizierte HW-Kennung des Geräts existiert nicht.
	`ERuntimeErrorCode.WrongModuleType`	Die spezifizierte HW-Kennung ist nicht die eines dezentralen Geräts.
	`ERuntimeErrorCode.WrongModuleState`	Das Gerät mit der spezifizierten HW-Kennung meldet bereits den Zustand Fault/Return.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.6.9 Ereignisse für IInstances

7.6.9.1 Ereignisse für Betriebszustand und Zykluskontrolle

Ereignisse für Betriebszustand und Zykluskontrolle

Für die Schnittstelle IInstances werden folgende Ereignisse ausgelöst:

Tabelle 7- 280 Ereignisse für IInstances

Ereignis	Ursache
OnOperatingStateChanged (Seite 278)	Der Betriebszustand des virtuellen Controllers hat sich geändert.
OnLedChanged (Seite 281)	Die LED-Anzeige des virtuellen Controllers hat sich geändert.
OnConfigurationChanging (Seite 283)	Die Konfiguration des virtuellen Controllers ändert sich: • Beim Hochlauf aus der Virtual SIMATIC Memory Card • Zu Beginn eines Downloads Wenn dieses Ereignis ausgelöst wird, wird die gespeicherte Variablentabelle zurückgesetzt.
OnConfigurationChanged (Seite 286)	Die Konfiguration des virtuellen Controllers hat sich geändert: • Nach dem Hochlauf aus der Virtual SIMATIC Memory Card • Am Ende eines Downloads • Wenn sich die IP-Adresse ändert
OnSyncPointReached (Seite 288)	Der virtuelle Controller hat einen Synchronisationspunkt erreicht. Wenn der virtuelle Controller in der Betriebsart `Default` betrieben wird, muss das Flag `SendSyncEventInDefaultMode` gesetzt werden, um das Ereignis zu empfangen. Siehe SendSyncEventInDefaultMode (Seite 248).

Ereignisse OnOperatingStateChanged

OnOperatingStateChanged

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 281 OnOperatingStateChanged - .NET (C#)

Syntax	`event Delegate_II_EREC_DT_EOS_EOS OnOperatingStateChanged;`
Parameter	Keine. Siehe Delegate_II_EREC_DT_EOS_EOS (Seite 334).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnOperatingStateChangedCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 282 RegisterOnOperatingStateChangedCallback() - Native C++

Syntax	`void RegisterOnOperatingStateChangedCallback(`
	`  EventCallback_II_SREC_ST_SROS_SROS in_CallbackFunction`
	`);`

Parameter	• `EventCallback_II_SREC_ST_SROS_SROS in_CallbackFunction:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_SROS_SROS (Seite 321).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

RegisterOnOperatingStateChangedEvent()

Wenn das Ereignis eintritt, wird das registrierte Event-Objekt in den signalisierten Zustand gesetzt. Es kann nur ein Event-Objekt für das Ereignis registriert sein. Das Registrieren eines neuen Event-Objekts führt zum Löschen des vorhergehenden.

Tabelle 7- 283 RegisterOnOperatingStateChangedEvent() - Native C++

Syntax	`void RegisterOnOperatingStateChangedEvent();`
	`void RegisterOnOperatingStateChangedEvent(`
	`  HANDLE* in_Event`
	`);`

Parameter	• `None:` Ein internes Event-Objekt wird registriert. • `HANDLE* in_Event:` Ein Handle zu einem anwenderspezifischen Event-Objekt. Das Event-Objekt wird registriert.
Rückgabewerte	Keine

Beispiel C++	`// Thread 1 ----------`
	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	`result = api->RegisterInstance(&psa);`
	`}`
	
	`// Register the internal event object`
	`psa->RegisterOnOperatingStateChangedEvent();`
	
	`// Thread 2 ----------`
	`while (condition)`
	`{`
	`// Wait for the event to be set (timeout after 10s)`
	`bool isEventSet = psa->WaitForOnOperat-`
	`ingStateChangedEvent(10000);`
	`if (isEventSet)`
	`{`
	`// Do Something`
	`…`
	`}`
	`}`

Beispiel C++	`// Thread 1 ----------`
	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	`result = api->RegisterInstance(&psa);`
	`}`
	
	`// Create an event object`
	`HANDLE eventHandle = CreateEvent(NULL, FALSE, FALSE, NULL);`
	
	`// Register the user created event object`
	`psa->RegisterOnOperatingStateChangedEvent(&eventHandle);`
	
	`// Do Something`
	`…`
	`// Clean up the handle`
	`CloseHandle(eventHandle);`
	
	`// Thread 2 ----------`
	`while (condition)`
	`{`
	`// Wait for the event to be set //OR:`
	`WaitForSingleObject(eventHandle, INFINITE); //psa->WaitForOnOper-`
	`atingStateChangedEvent();`
	
	`// Do Something`
	`…`
	`}`

UnregisterOnOperatingStateChangedCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 284 UnregisterOnOperatingStateChangedCallback() - Native C++

Syntax	`void UnregisterOnOperatingStateChangedCallback();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnOperatingStateChangedEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 285 UnregisterOnOperatingStateChangedEvent() - Native C++

Syntax	`void UnregisterOnOperatingStateChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

Tabelle 7- 286 UnregisterOnOperatingStateChangedEvent() - .NET (C#)

Syntax	`void UnregisterOnOperatingStateChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

WaitForOnOperatingStateChangedEvent()

Die Funktion blockiert das Programm solange, bis das registrierte Event-Objekt im signalisierten Zustand ist oder bis das Timeout-Intervall überschritten ist.

Tabelle 7- 287 WaitForOnOperatingStateChangedEvent() - Native C++

Syntax	`bool WaitForOnOperatingStateChangedEvent();`
	`bool WaitForOnOperatingStateChangedEvent(`
	`  UINT32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UINT32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Tabelle 7- 288 WaitForOnOperatingStateChangedEvent() - .NET (C#)

Syntax	`bool WaitForOnOperatingStateChangedEvent();`
	`bool WaitForOnOperatingStateChangedEvent(`
	`  UInt32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UInt32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Ereignisse OnLedChanged

OnLedChanged

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 289 OnLedChanged - .NET (C#)

Syntax	`event Delegate_II_EREC_DT_ELT_ELM OnLedChanged;`
Parameter	Keine. Siehe Delegate_II_EREC_DT_ELT_ELM (Seite 335).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnLedChangedCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 290 RegisterOnLedChangedCallback() - Native C++

Syntax	`void RegisterOnLedChangedCallback(`
	`  EventCallback_II_SREC_ST_SRLT_SRLM in_CallbackFunction`
	`);`

Parameter	• `EventCallback_II_SREC_ST_SRLT_SRLM in_CallbackFunction:` Eine Callback-Funktion, die ein Ereignis abonniert. Siehe EventCallback_II_SREC_ST_SRLT_SRLM (Seite 324).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

RegisterOnLedChangedEvent()

Wenn das Ereignis eintritt, wird das registrierte Event-Objekt in den signalisierten Zustand gesetzt. Es kann nur ein Event-Objekt für das Ereignis registriert sein. Das Registrieren eines neuen Event-Objekts führt zum Löschen des vorhergehenden.

Tabelle 7- 291 RegisterOnLedChangedEvent() - Native C++

Syntax	`void RegisterOnLedChangedEvent();`
	`void RegisterOnLedChangedEvent(`
	`  HANDLE* in_Event`
	`);`

Parameter	• `None:` Ein internes Event-Objekt wird registriert. • `HANDLE* in_Event:` Ein Handle zu einem anwenderspezifischen Event-Objekt. Das Event-Objekt wird registriert.
Rückgabewerte	Keine

UnregisterOnLedChangedCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 292 UnregisterOnLedChangedCallback() - Native C++

Syntax	`void UnregisterOnLedChangedCallback();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnLedChangedEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 293 UnregisterOnLedChangedEvent() - Native C++

Syntax	`void UnregisterOnLedChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

Tabelle 7- 294 UnregisterOnLedChangedEvent() - .NET (C#)

Syntax	`void UnregisterOnLedChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

WaitForOnLedChangedEvent()

Die Funktion blockiert das Programm solange, bis das registrierte Event-Objekt im signalisierten Zustand ist oder bis das Timeout-Intervall überschritten ist.

Tabelle 7- 295 WaitForOnLedChangedEvent() - Native C++

Syntax	`bool WaitForOnLedChangedEvent();`
	`bool WaitForOnLedChangedEvent(`
	`  UINT32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UINT32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Tabelle 7- 296 WaitForOnLedChangedEvent() - .NET (C#)

Syntax	`bool WaitForOnLedChangedEvent();`
	`bool WaitForOnLedChangedEvent(`
	`  UInt32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UInt32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Ereignisse OnConfigurationChanging

OnConfigurationChanging

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 297 OnConfigurationChanging - .NET (C#)

Syntax	`event Delegate_II_EREC_DT OnConfigurationChanging;`
Parameter	Keine. Siehe Delegate_II_EREC_DT (Seite 334).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnConfigurationChangingCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 298 RegisterOnConfigurationChangingCallback() - Native C++

Syntax	`void RegisterOnConfigurationChangingCallback( EventCallback_II_SREC_ST in_CallbackFunction );`
Parameter	• `EventCallback_II_SREC_ST in_CallbackFunction:` Eine Callback-Funktion, die ein Ereignis abonniert. Siehe EventCallback_II_SREC_ST (Seite 323).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

RegisterOnConfigurationChangingEvent()

Wenn das Ereignis eintritt, wird das registrierte Event-Objekt in den signalisierten Zustand gesetzt. Es kann nur ein Event-Objekt für das Ereignis registriert sein. Die Registrierung eines neuen Event-Objekts führt zum Löschen des vorhergehenden.

Tabelle 7- 299 RegisterOnConfigurationChangingEvent() - Native C++

Syntax	`void RegisterOnConfigurationChangingEvent();` `void RegisterOnConfigurationChangingEvent( HANDLE* in_Event );`
Parameter	• `None:` Ein internes Event-Objekt wird registriert. • `HANDLE* in_Event:` Ein Handle zu einem anwenderspezifischen Event-Objekt. Das Event-Objekt wird registriert.
Rückgabewerte	Keine

UnregisterOnConfigurationChangingCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 300 UnregisterOnConfigurationChangingCallback() - Native C++

Syntax	`void UnregisterOnConfigurationChangingCallback();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnConfigurationChangingEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 301 UnregisterOnConfigurationChangingEvent() - Native C++

Syntax	`void UnregisterOnConfigurationChangingEvent();`
Parameter	Keine
Rückgabewerte	Keine

Tabelle 7- 302 UnregisterOnConfigurationChangingEvent() - .NET (C#)

Syntax	`void UnregisterOnConfigurationChangingEvent();`
Parameter	Keine
Rückgabewerte	Keine

WaitForOnConfigurationChangingEvent()

Die Funktion blockiert das Programm solange, bis das registrierte Event-Objekt im signalisierten Zustand ist oder bis das Timeout-Intervall überschritten ist.

Tabelle 7- 303 WaitForOnConfigurationChangingEvent() - Native C++

Syntax	`bool WaitForOnConfigurationChangingEvent();`
	`bool WaitForOnConfigurationChangingEvent(`
	`  UINT32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UINT32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Tabelle 7- 304 WaitForOnConfigurationChangingEvent() - .NET (C#)

Syntax	`bool WaitForOnConfigurationChangingEvent();`
	`bool WaitForOnConfigurationChangingEvent(`
	`  UInt32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UInt32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true`: Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false`: Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Ereignisse OnConfigurationChanged

OnConfigurationChanged

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 305 OnConfigurationChanged - .NET (C#)

Syntax	`event Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32 OnConfigurationChanged;`
Parameter	Keine. Siehe Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32 (Seite 337).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnConfigurationChangedCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 306 RegisterOnConfigurationChangedCallback() - Native C++

Syntax	`void RegisterOnConfigurationChangedCallback(`
	`  EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32`
	`in_CallbackFunction`
	`);`

Parameter	• `EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32 in_CallbackFunction:` Eine Callback-Funktion, die ein Ereignis abonniert. Siehe EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32 (Seite 323).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

RegisterOnConfigurationChangedEvent()

Wenn das Ereignis eintritt, wird das registrierte Event-Objekt in den signalisierten Zustand gesetzt. Es kann nur ein Event-Objekt für das Ereignis registriert sein. Das Registrieren eines neuen Event-Objekts führt zum Löschen des vorhergehenden.

Tabelle 7- 307 RegisterOnConfigurationChangedEvent() - Native C++

Syntax	`void RegisterOnConfigurationChangedEvent();`
	`void RegisterOnConfigurationChangedEvent(`
	`  HANDLE* in_Event`
	`);`

Parameter	• `None:` Ein internes Event-Objekt wird registriert. • `HANDLE* in_Event:` Ein Handle zu einem anwenderspezifischen Event-Objekt. Das Event-Objekt wird registriert.
Rückgabewerte	Keine

UnregisterOnConfigurationChangedCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 308 UnregisterOnConfigurationChangedCallback() - Native C++

Syntax	`void UnregisterOnConfigurationChangedCallback();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnConfigurationChangedEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 309 UnregisterOnConfigurationChangedEvent() - Native C++

Syntax	`void UnregisterOnConfigurationChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

Tabelle 7- 310 UnregisterOnConfigurationChangedEvent() - .NET (C#)

Syntax	`void UnregisterOnConfigurationChangedEvent();`
Parameter	Keine
Rückgabewerte	Keine

WaitForOnConfigurationChangedEvent()

Die Funktion blockiert das Programm solange, bis das registrierte Event-Objekt im signalisierten Zustand ist oder bis das Timeout-Intervall überschritten ist.

Tabelle 7- 311 WaitForOnConfigurationChangedEvent() - Native C++

Syntax	`bool WaitForOnConfigurationChangedEvent();`
	`bool WaitForOnConfigurationChangedEvent(`
	`  UINT32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UINT32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Tabelle 7- 312 WaitForOnConfigurationChangedEvent() - .NET (C#)

Syntax	`bool WaitForOnConfigurationChangedEvent();`
	`bool WaitForOnConfigurationChangedEvent(`
	`  UInt32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UInt32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Ereignisse OnSyncPointReached

OnSyncPointReached

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 313 OnSyncPointReached - .NET (C#)

Syntax	`event Delegate_II_EREC_DT_UINT32_INT64_INT64_UINT32 OnSyncPoin-tReached;`
Parameter	Keine. Siehe Delegate_II_EREC_DT_UINT32_INT64_INT64_UINT32 (Seite 336).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnSyncPointReachedCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 314 RegisterOnSyncPointReachedCallback() - Native C++

Syntax	`void RegisterOnSyncPointReachedCallback( EventCallback_II_SREC_ST_UINT32_INT64_INT64_UINT32 in_Callback-Function );`
Parameter	• `EventCallback_II_SREC_ST_UINT32_INT64_INT64_UINT32 in_Callback-Function:` Eine Callback-Funktion, die ein Ereignis abonniert. Siehe EventCallback_II_SREC_ST_UINT32_INT64_INT64_UINT32 (Seite 322).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

RegisterOnSyncPointReachedEvent()

Wenn das Ereignis eintritt, wird das registrierte Event-Objekt in den signalisierten Zustand gesetzt. Es kann nur ein Event-Objekt für das Ereignis registriert sein. Das Registrieren eines neuen Event-Objekts führt zum Löschen des vorhergehenden.

Tabelle 7- 315 RegisterOnSyncPointReachedEvent() - Native C++

Syntax	`void RegisterOnSyncPointReachedEvent();` `void RegisterOnSyncPointReachedEvent( HANDLE* in_Event );`
Parameter	• `None:` Ein internes Event-Objekt wird registriert. • `HANDLE* in_Event:` Ein Handle zu einem anwenderspezifischen Event-Objekt. Das Event-Objekt wird registriert.
Rückgabewerte	Keine

UnregisterOnSyncPointReachedCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 316 UnregisterOnSyncPointReachedCallback() - Native C++

Syntax	`void UnregisterOnSyncPointReachedCallback();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnSyncPointReachedEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 317 UnregisterOnSyncPointReachedEvent() - Native C++

Syntax	`void UnregisterOnSyncPointReachedEvent();`
Parameter	Keine
Rückgabewerte	Keine

Tabelle 7- 318 UnregisterOnSyncPointReachedEvent() - .NET (C#)

Syntax	`void UnregisterOnSyncPointReachedEvent();`
Parameter	Keine
Rückgabewerte	Keine

WaitForOnSyncPointReachedEvent()

Die Funktion blockiert das Programm solange, bis das registrierte Event-Objekt im signalisierten Zustand ist oder bis das Timeout-Intervall überschritten ist.

Tabelle 7- 319 WaitForOnSyncPointReachedEvent() - Native C++

Syntax	`SOnSyncPointReachedResult WaitForOnSyncPointReachedEvent();`
	`SOnSyncPontReachedResult WaitForOnEndOfCycleOnSyncPoin-tReachedEvent(`
	`  UINT32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UINT32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `SOnSyncPointReachedResult`: Eine Struktur, die Informationen zum Ereignis liefert. Siehe SOnSyncPointReachedResult (Seite 360).

Tabelle 7- 320 WaitForOnSyncPointReachedEvent() - .NET (C#)

Syntax	`SOnSyncPointReachedResult WaitForOnSyncPointReachedEvent();`
	`SOnSyncPointReachedResult WaitForOnSyncPointReachedEvent(`
	`  UInt32 in_Time_ms`
	`);`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UInt32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `SOnSyncPointReachedResult`: Eine Struktur, die Informationen zum Ereignis liefert. Siehe SOnSyncPointReachedResult (Seite 360).

7.6.9.2 Ereignisse für Azyklische Dienste

Ereignisse OnDataRecordRead / OnDataRecordWrite

OnDataRecordRead

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 321 OnDataRecordRead - .NET (C#)

Syntax	`event Delegate_II_EREC_DT_SDRI OnDataRecordRead;`
Parameter	Keine. Siehe Delegate_II_EREC_DT_SDRI (Seite 339).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

OnDataRecordWrite

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 322 OnDataRecordWrite - .NET (C#)

Syntax	`event Delegate_II_EREC_DT_SDR OnDataRecordWrite;`
Parameter	Keine. Siehe Delegate_II_EREC_DT_SDR (Seite 338).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnDataRecordReadCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 323 RegisterOnDataRecordReadCallback() - Native C++

Syntax	`void RegisterOnDataRecordReadCallback (`
	`  Event Callback_II_SREC_ST_SDRI in_CallbackFunction`
	`);`

Parameter	• `EventCallback_II_SREC_ST_SDRI in_CallbackFunction:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_SDRI.
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnDataRecordReadCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 324 UnregisterOnDataRecordReadCallback() - Native C++

Syntax	`void UnregisterOnDataRecordReadCallback();`
Parameter	Keine
Rückgabewerte	Keine

RegisterOnDataRecordWriteCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 325 RegisterOnDataRecordWriteCallback() - Native C++

Syntax	`void RegisterOnDataRecordWriteCallback (`
	`  EventCallback_II_SREC_ST_SDRI_BYTE in_CallbackFunction`
	`);`

Parameter	• `EventCallback_II_SREC_ST_SDRI_BYTE in_CallbackFunction:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_SDRI_BYTE.
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnDataRecordWriteCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 326 UnregisterOnDataRecordWriteCallback() - Native C++

Syntax	`void UnregisterOnDataRecordWriteCallback();`
Parameter	Keine
Rückgabewerte	Keine

Ereignisse OnAlarmNotification

OnAlarmNotificationDone()

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 327 OnAlarmNotificationDone() - .NET (C#)

Syntax	`event Delegate_SREC_ST_UINT32_UINT32 OnAlarmNotificationDone;`
Parameter	Keine. Siehe Delegate_SREC_ST_UINT32_UINT32 (Seite 343).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnAlarmNotificationDoneCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 328 RegisterOnAlarmNotificationDoneCallback() - Native C++

Syntax	`void RegisterOnAlarmNotificationDoneCallback (`
	`  Event Callback_II_SREC_ST_SDRI in_CallbackFunction`
	`);`

Parameter	• `EventCallback_II_SREC_ST_UINT32_UINT32 in_CallbackFunction:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_UINT32_UINT32 (Seite 327).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnAlarmNotificationDoneCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 329 UnregisterOnAlarmNotificationDoneCallback() - Native C++

Syntax	`void UnregisterOnAlarmNotificationDoneCallback ();`
Parameter	Keine
Rückgabewerte	Keine

Ereignisse OnProcessEvent

OnProcessEventDone()

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 330 OnProcessEventDone() - .NET (C#)

Syntax	`event Delegate_SREC_ST_UINT32_UINT32_EPET_UINT32 OnProcessEvent-Done;`
Parameter	Keine. Siehe Delegate_SREC_ST_UINT32_UINT32_EPET_UINT32. (Seite 341)
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnProcessEventDoneCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 331 RegisterOnProcessEventDoneCallback() - Native C++

Syntax	• `void RegisterOnProcessEventDoneCallback ( EventCallback_II_SREC_ST_UINT32_UINT32_EPET_UINT32 in_CallbackFunction );`
Parameter	• `EventCallback_II_SREC_ST_UINT32_UINT32_EPET_UINT32 in_Callback-Function:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_UINT32_UINT32_EPET_UINT32 (Seite 328)
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnProcessEventDoneCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 332 UnregisterOnProcessEventDoneCallback() - Native C++

Syntax	`void UnregisterOnProcessEventDoneCallback ();`
Parameter	Keine
Rückgabewerte	Keine

Ereignisse OnPullOrPlugEvent

OnPullOrPlugEventDone()

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 333 OnPullOrPlugEventDone() - .NET (C#)

Syntax	`event Delegate_SREC_ST_UINT32_EPPET_UINT32 OnPullOrPlugEventDone;`
Parameter	Keine. Siehe Delegate_SREC_ST_UINT32_EPPET_UINT32 (Seite 340).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnPullOrPlugEventDoneCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 334 RegisterOnPullOrPlugEventDoneCallback() - Native C++

Syntax	• `void RegisterOnPullOrPlugEventDoneCallback ( EventCallback_II_SREC_ST_UINT32_EPPET_UINT32 in_CallbackFunc-tion );`
Parameter	• `EventCallback_II_SREC_ST_UINT32_EPPET_UINT32 in_CallbackFunc-tion:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_UINT32_EPPET_UINT32 (Seite 329).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnPullOrPlugEventDoneCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 335 UnregisterOnPullOrPlugEventDoneCallback() - Native C++

Syntax	`void UnregisterOnPullOrPlugEventDoneCallback ();`
Parameter	Keine
Rückgabewerte	Keine

Ereignisse OnStatusEvent

OnStatusEventDone()

Meldet eine Event-Handler Methode an- oder ab.

Tabelle 7- 336 OnStatusEventDone() - .NET (C#)

Syntax	`event Delegate_SREC_ST_UINT32 OnStatusEventDone;`
Parameter	Keine. Siehe Delegate_SREC_ST_UINT32 (Seite 342).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnStatusEventDoneCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 337 RegisterOnStatusEventDoneCallback() - Native C++

Syntax	• `void RegisterOnStatusEventDoneCallback ( EventCallback_II_SREC_ST_UINT32 in_CallbackFunction );`
Parameter	• `EventCallback_II_SREC_ST_UINT32 in_CallbackFunction:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_UINT32 (Seite 331).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnStatusEventDoneCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 338 UnregisterOnStatusEventDoneCallback() - Native C++

Syntax	`void UnregisterOnStatusEventDoneCallback ();`
Parameter	Keine
Rückgabewerte	Keine

Ereignisse OnProfileEvent

OnProfileEventDone()

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 339 OnProfileEventDone() - .NET (C#)

Syntax	`event Delegate_SREC_ST_UINT32 OnProfileEventDone;`
Parameter	Keine. Siehe Delegate_SREC_ST_UINT32 (Seite 342).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnProfileEventDoneCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 340 RegisterOnProfileEventDoneCallback() - Native C++

Syntax	• `void RegisterOnProfileEventDoneCallback ( EventCallback_II_SREC_ST_UINT32 in_CallbackFunction );`
Parameter	• `EventCallback_II_SREC_ST_UINT32 in_CallbackFunction:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_UINT32 (Seite 331).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnProfileEventDoneCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 341 UnregisterOnProfileEventDoneCallback() - Native C++

Syntax	`void UnregisterOnProfileEventDoneCallback ();`
Parameter	Keine
Rückgabewerte	Keine

Ereignisse OnUpdateEvent

OnUpdateEventDone()

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 342 OnUpdateEventDone() - .NET (C#)

Syntax	`event Delegate_SREC_ST_UINT32 OnUpdateEventDone;`
Parameter	Keine. Siehe Delegate_SREC_ST_UINT32 (Seite 342).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnUpdateEventDoneCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 343 RegisterOnUpdateEventDoneCallback() - Native C++

Syntax	• `void RegisterOnUpdateEventDoneCallback ( EventCallback_II_SREC_ST_UINT32 in_CallbackFunction );`
Parameter	• `EventCallback_II_SREC_ST_UINT32 in_CallbackFunction:` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_UINT32 (Seite 331).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnUpdateEventDoneCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 344 UnregisterOnUpdateEventDoneCallback() - Native C++

Syntax	`void UnregisterOnUpdateEventDoneCallback ();`
Parameter	Keine
Rückgabewerte	Keine

Ereignisse RackOrStationFault

OnRackOrStationFaultEvent

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 345 OnRackOrStationFaultEvent - .NET (C#)

Syntax	`event Delegate_SREC_ST_UINT32_ERSFET OnRackOrStationFault;`
Parameter	Keine. Siehe Delegate_SREC_ST_UINT32_ERSFET (Seite 344).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnRackOrStationFaultEventCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Löschen der vorhergehenden.

Tabelle 7- 346 RegisterOnRackOrStationFaultEventCallback() - Native C++

Syntax	`void RegisterOnRackOrStationFaultEventCallback ( EventCallback_II_SREC_ST_UINT32_ERSFET in_CallbackFunction );`
Parameter	• `EventCallback_II_ SREC_ST_UINT32_ERSFET in_CallbackFunction.` Eine Callback-Funktion, die das Ereignis abonniert. Siehe EventCallback_II_SREC_ST_UINT32_ERSFET (Seite 330)
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

UnregisterOnRackOrStationFaultEventCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 347 UnregisterOnRackOrStationFaultEventCallback() - Native C++

Syntax	`void UnregisterOnRackOrStationFaultEventCallback ();`
Parameter	Keine
Rückgabewerte	Keine

7.7 API IRemoteRuntimeManager

7.7.1 Schnittstellen - Information und Einstellungen

Dispose()

Löscht die managed Schnittstelle und entlädt die nativen Komponenten der Anwenderschnittstellen.

Hinweis

Wenn die Schnittstelle des Remote Runtime Managers gelöscht wird, dann kann keine IInstance-Schnittstelle, die von der IRemoteRuntimeManager-Schnittstelle erzeugt wurde, mehr genutzt werden.

Der .NET Garbage Collector bereinigt Ihre IRemoteRuntimeManager-Schnittstelle, wenn keine aktive Referenz mehr vorhanden ist.

Tabelle 7- 348 Dispose() - .NET (C#)

Syntax	`void Dispose()`
Parameter	Keine
Rückgabewerte	Keine

GetVersion()

Liefert die Version des Remote Runtime Managers zurück. Wenn die Funktion fehlschlägt, wird die Version 0.0 zurückgegeben.

Tabelle 7- 349 GetVersion() - Native C++

Syntax	`UINT32 GetVersion();`
Parameter	Keine
Rückgabewerte	`UINT32`: Remote Runtime Manager Version (`HIWORD` = Major, `LOWORD` = Minor)

Tabelle 7- 350 Version { get; } - .NET (C#)

Syntax	`UInt32 Version { get; }`
Parameter	Keine
Rückgabewerte	`UInt32`: Remote Runtime Manager Version (`HIWORD` = Major, `LOWORD` = Minor)

GetIP() / IP { get; }

Liefert die IP-Adresse des PC, auf dem der Remote Runtime Manager läuft. Wenn die Funktion fehlschlägt, ist der Rückgabewert 0.

Tabelle 7- 351 GetIP() - Native C++

Syntax	`UIP GetIP();`
Parameter	Keine
Rückgabewerte	`UIP`: Liefert die IP-Adresse des PC, auf dem der Runtime Manager läuft.

Tabelle 7- 352 IP { get; } - .NET (C#)

Syntax	`SIP IP { get; }`
Parameter	Keine
Rückgabewerte	`SIP`: Liefert die IP-Adresse des PC, auf dem der Runtime Manager läuft.

GetPort() / Port { get; }

Liefert den offenen Port des PC, auf dem der Remote Runtime Manager läuft. Wenn die Funktion fehlschlägt, ist der Rückgabewert 0.

Tabelle 7- 353 GetPort() - Native C++

Syntax	`UINT16 GetPort();`
Parameter	Keine
Rückgabewerte	`UINT16`: Offener Port des PC, auf dem der Remote Runtime Manager läuft.

Tabelle 7- 354 Port { get; } - .NET (C#)

Syntax	`UInt16 Port { get; }`
Parameter	Keine
Rückgabewerte	`UInt16`: Offener Port des PC, auf dem der Remote Runtime Manager läuft.

GetRemoteComputerName() / RemoteComputerName { get; }

Liefert den Namen des PC, auf dem der Remote Runtime Manager läuft.

Tabelle 7- 355 GetRemoteComputerName() - Native C++

Syntax	`ERuntimeErrorCode GetRemoteComputerName(`
	`  WCHAR* inout_Name,`
	`  UINT32 in_ArrayLength`
	`);`	

Parameter	• `WCHAR* inout_Name:` Ein benutzerallokiertes Feld für den Computernamen. • `UINT32 in_ArrayLength:` Die Feldgröße. Das Feld sollte größer sein als `MAX_COMPUTERNAME_LENGTH`.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`SREC_INDEX_OUT_OF_RANGE`	Das Feld ist zu klein, um den Computernamen zu empfangen.

Tabelle 7- 356 RemoteComputerName { get; } - .NET (C#)

Syntax	`string RemoteComputerName { get; }`	
Parameter	Keine	
Rückgabewerte	`string`: Name des PC, auf dem der Remote Runtime Manager läuft.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`ERuntimeErrorCode.IndexOutOfRange`	Das Feld ist zu klein, um den Computernamen zu empfangen.

Disconnect()

Schließt die Verbindung zum Remote Runtime Manager.

Hinweis

Alle Anwendungen, die mit dem Remote Runtime Manager verbunden sind, verlieren diese Verbindung.

Tabelle 7- 357 Disconnect() - Native C++

Syntax	`ERuntimeErrorCode Disconnect();`	
Parameter	Keine	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.

Tabelle 7- 358 Disconnect() - .NET (C#)

Syntax	`void Disconnect();`	
Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

7.7.2 Simulation Runtime Instanzen

7.7.2.1 Simulation Runtime Instanzen (Remote)

GetRegisteredInstancesCount()

Liefert die Anzahl der Instanzen zurück, die im Runtime Manager registriert sind. Wenn die Funktion fehlschlägt, ist der Rückgabewert 0.

Tabelle 7- 359 GetRegisteredInstancesCount() - Native C++

Syntax	`UINT32 GetRegisteredInstancesCount();`
Parameter	Keine
Rückgabewerte	`UINT32`: Anzahl der verfügbaren Instanzen.

GetRegisteredInstanceInfoAt()

Liefert die Information über eine bereits registrierte Instanz zurück.

Sie können die ID oder den Namen verwenden, um eine Schnittstelle dieser Instanz zu erzeugen (siehe `CreateInterface()`).

Tabelle 7- 360 GetRegisteredInstanceInfoAt() - Native C++

Syntax	`ERuntimeErrorCode GetRegisteredInstanceInfoAt(`
	`  UINT32 in_Index,`
	`  SInstanceInfo* out_InstanceInfo`
	`);`	

Parameter	• `UINT32 in_Index:` Index der erzeugten Instanz, von der Sie die Information empfangen möchten. Der Index muss kleiner sein als der Wert, den Sie empfangen, wenn Sie `GetRegisteredInstanceCount()` aufrufen. • `SInstanceInfo* out_InstanceInfo:` Die Information mit Name und ID der Instanz. Siehe SInstanceInfo (Seite 354).	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_INDEX OUT_OF_RANGE`	Es gibt keine Instanz-Information zu diesem Index.

RegisteredInstanceInfo { get; }

Liefert die Information über eine bereits registrierte Instanz zurück. Sie können die ID oder den Namen dieser Instanz verwenden, um eine Schnittstelle von dieser Instanz zu erzeugen, siehe `CreateInterface()`.

Tabelle 7- 361 RegisterInstanceInfo { get; } - .NET (C#)

Syntax	`SInstanceInfo[] RegisteredInstanceInfo { get; }`	
Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

RegisterInstance()

Registriert im Runtime Manager eine neue Instanz eines virtuellen Controllers. Erzeugt und liefert eine Schnittstelle von dieser Instanz zurück.

Tabelle 7- 362 RegisterInstance() - Native C++

Syntax	`ERuntimeErrorCode RegisterInstance(`
	`  IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode RegisterInstance(`
	`  WCHAR* in_InstanceName,`
	`  IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode RegisterInstance(`
	`  ECPUType in_CPUType,`
	`  IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode RegisterInstance(`
	`  ECPUType in_CPUType,`
	`  WCHAR* in_InstanceName,`
	`  IInstance** out_InstanceInterface`
	`);`	

Parameter	• `ECPUType in_CPUType:` Definiert, welcher CPU-Typ beim Start der Instanz simuliert wird. Die Voreinstellung ist `"SRCT_1500_Unspecified"`. Wenn über STEP 7 oder von der Virtual SIMATIC Memory Card ein anderer CPU-Typ geladen wird, dann gilt dieser CPU-Typ. • `WCHAR* in_InstanceName:` Name, den die Instanz erhalten soll. Jede Instanz muss einen eindeutigen Namen erhalten. Wenn kein Name vergeben wird beim Registrieren einer neuen Instanz, dann erhält die Instanz den Namen `"Instance_#"` (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name `"Instance_#.#"` verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als `DINSTANCE_NAME_LENGTH`. Siehe Datentypen (Seite 344). • `IInstance** out_InstanceInterface:` Zeiger auf einen Simulation Runtime Schnittstellenzeiger. Der Zeiger **muss** mit `NULL` initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Name oder der IInstance-Zeiger ist ungültig.
	`SREC_LIMIT_REACHED`	Es sind bereits 16 Instanzen im Runtime Manager registriert.
	`SREC_ALREADY_EXISTS`	Eine Instanz mit diesem Namen existiert bereits.

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`// Example: How To Create And Register An Instance`
	`// And To Get An Interface Of The Instance The Same Time`
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	` result = api->RegisterInstance(&psa);`
	`}`

Hinweis

Native C++

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118).

Tabelle 7- 363 RegisterInstance() - .NET (C#)

Syntax	`IInstance RegisterInstance();`
	`IInstance RegisterInstance(`
	` string in_InstanceName`
	`);`
	`IInstance RegisterInstance(`
	` ECPUType in_CPUType`
	`);`
	`IInstance RegisterInstance(`
	` ECPUType in_CPUType`
	` string in_InstanceName`
	`);`	

Parameter	• `ECPUType in_CPUType:` Definiert, welcher CPU-Typ beim Start der Instanz simuliert wird. Die Voreinstellung ist `"ECPUType.Unspecified"`. Wenn über STEP 7 oder von der Virtual SIMATIC Memory Card ein anderer CPU-Typ geladen wird, dann gilt dieser CPU-Typ. • `string in_InstanceName:` Name, den die Instanz erhalten soll. Jede Instanz muss einen eindeutigen Namen erhalten. Wenn kein Name vergeben wird beim Registrieren einer neuen Instanz, dann erhält die Instanz den Namen `"Instance_#"` (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name `"Instance_#.#"` verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als `DINSTANCE_NAME_LENGTH`. Siehe Datentypen (Seite 344).	
Rückgabewerte	Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers. Ansonsten einen Null-Zeiger.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Name ist ungültig.

	`ERuntimeErrorCode.LimitReached`	Es sind bereits 16 Instanzen im Runtime Manager registriert.
	`ERuntimeErrorCode.AlreadyExists`	Eine Instanz mit diesem Namen existiert bereits.

RegisterCustomInstance()

Registriert im Runtime Manager eine neue Instanz eines virtuellen Controllers. Erzeugt und liefert eine Schnittstelle von dieser Instanz zurück.

Tabelle 7- 364 RegisterCustomInstance() - Native C++

Syntax	`ERuntimeErrorCode RegisterCustomInstance(`
	` WCHAR* in_VplcDll,`
	` IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode RegisterCustomInstance(`
	` WCHAR* in_VplcDll,`
	` WCHAR* in_InstanceName,`
	` IInstance** out_InstanceInterface`
	`);`

Parameter	• `WCHAR* in_VplcDll:` Der vollständige Pfad zur DLL des virtuellen Controllers, den die Siemens.Simatic.Simulation.Runtime.Instance.exe bei PowerOn laden wird. • `WCHAR* in_InstanceName:` Name, den die Instanz erhalten soll. Jede Instanz muss einen eindeutigen Namen erhalten. Wenn kein Name vergeben wird beim Registrieren einer neuen Instanz, dann erhält die Instanz den Namen `"Instance_#"` (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name `"Instance_#.#"` verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als `DINSTANCE_NAME_LENGTH`. Siehe Datentypen (Seite 344). • `IInstance** out_InstanceInterface:` Zeiger auf einen Simulation Runtime Schnittstellenzeiger. Der Zeiger muss mit `NULL` initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt.

Rückgabewerte	Runtime Fehlercode	Bedingung
	SREC_OK	Die Funktion ist erfolgreich.
	SREC_INTERFACE_REMOVED	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	SREC_TIMEOUT	Die Funktion kehrt nicht rechtzeitig wieder.
	SREC_WRONG_ARGUMENT	Der DLL-Name, der Instanzname oder der IInstance-Zeiger ist ungültig.
	SREC_LIMIT_REACHED	Es sind bereits 16 Instanzen im Runtime Manager registriert.
	SREC_ALREADY_EXISTS	Eine Instanz mit diesem Namen existiert bereits.

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`// Example: How To Create And Register An Instance`
	`// And To Get An Interface Of The Instance The Same Time`
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	`result = api->RegisterCustomInstance("C:\\Temp\\vplc.dll");`
	`}`	

Hinweis

Native C++

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118).

Tabelle 7- 365 RegisterCustomInstance() - .NET (C#)

Syntax	`IInstance RegisterCustomInstance(`
	`string in_VplcDll`
	`);`
	`IInstance RegisterCustomInstance(`
	`string in_VplcDll,`
	`string in_InstanceName`
	`);`

Parameter	• `string in_VplcDll:` Der vollständige Pfad zur DLL des virtuellen Controllers, den die Siemens.Simatic.Simulation.Runtime.Instance.exe bei PowerOn laden wird. • `string in_InstanceName:` Name, den die Instanz erhalten soll. Jede Instanz muss einen eindeutigen Namen erhalten. Wenn kein Name vergeben wird beim Registrieren einer neuen Instanz, dann erhält die Instanz den Namen `"Instance_#"` (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name `"Instance_#.#"` verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als `DINSTANCE_NAME_LENGTH`. Siehe Datentypen (Seite 344).

Rückgabewerte	Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers, ansonsten ein Null-Zeiger.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Name oder die ID ist ungültig.
	`ERuntimeErrorCode.LimitReached`	Es sind bereits 16 Instanzen im Runtime Manager registriert.
	`ERuntimeErrorCode.AlreadyExists`	Eine Instanz mit diesem Namen existiert bereits.

CreateInterface()

Erzeugt und liefert eine Schnittstelle einer bereits registrierten Instanz eines virtuellen Controllers zurück.

Die Instanz kann über die Anwendung registriert worden sein oder über eine andere Anwendung, die die Simulation Runtime API nutzt.

Tabelle 7- 366 CreateInterface() - Native C++

Syntax	`ERuntimeErrorCode CreateInterface(`
	`  WCHAR* in_InstanceName,`
	`  IInstance** out_InstanceInterface`
	`);`
	`ERuntimeErrorCode CreateInterface(`
	`  INT32 in_InstanceID,`
	`  IInstance** out_InstanceInterface`
	`);`	

Parameter	• `INT32 in_InstanceID:` Die ID der registrierten Instanz, von der Sie die Schnittstelle emfangen möchten. • `WCHAR* in_InstanceName:` Der Name der registrierten Instanz, von der Sie die Schnittstelle emfangen möchten. • `IInstance** out_InstanceInterface:` Zeiger auf einen Simulation Runtime Schnittstellenzeiger. Der Zeiger muss mit `NULL` initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_INTERFACE_REMOVED`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`SREC_TIMEOUT`	Die Funktion kehrt nicht rechtzeitig wieder.
	`SREC_WRONG_ARGUMENT`	Der Name, die ID oder der IInstance-Zeiger ist ungültig.
	`SREC_DOES_NOT_EXIST`	Die Instanz ist nicht im Runtime Manager registriert.

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`IInstance* psa1 = NULL;`
	`IInstance* psa2 = NULL;`
	`if (result == SREC_OK)`
	`{`
	`result = api->CreateInterface(0, &psa1);`
	`result = api->CreateInterface(0, &psa2); // psa2 will be the same as psa1`
	`}`

Beispiel C++	`ISimulationRuntimeManager * api = NULL;`
	`ERuntimeErrorCode result = Initialize(&api);`
	
	`IInstance* psa = NULL;`
	`if (result == SREC_OK)`
	`{`
	`result = api->CreateInterface(L"My SimulationRuntime Instance", &psa);`
	`}`

Hinweis

Native C++

Wenn Sie die Schnittstelle nicht mehr benötigen, dann löschen Sie die Schnittstelle.

Siehe DestroyInterface() (Seite 118)

Tabelle 7- 367 CreateInterface() - .NET (C#)

Syntax	`IInstance CreateInterface(`
	`string in_InstanceName`
	`);`
	`IInstance CreateInterface(`
	`INT32 in_InstanceID`
	`);`	

Parameter	• `INT32 in_InstanceID:` Die ID der registrierten Instanz, von der Sie die Schnittstelle emfangen möchten. • `string in_InstanceName:` Der Name der registrierten Instanz, von der Sie die Schnittstelle emfangen möchten.	
Rückgabewerte	Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers, ansonsten ein Null-Zeiger.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.InterfaceRemoved`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeErrorCode.WrongArgument`	Der Name oder die ID ist ungültig.
	`ERuntimeErrorCode.DoesNotExists`	Die Instanz ist nicht im Runtime Manager registriert.

7.7.3 Ereignisse für IRemoteRuntimeManager

7.7.3.1 Ereignisse OnConnectionLost

Beschreibung

Das Ereignis wird ausgelöst, wenn die Verbindung zum Remote Runtime Manager gelöst wurde.

OnConnectionLost

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 368 OnConnectionLost - .NET (C#)

Syntax	`event Delegate_IRRTM OnConnectionLost;`
Parameter	Keine. Siehe Delegate_IRRTM (Seite 337)
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnConnectionLostCallback()

Wenn das Ereignis eintritt, wird die registrierte Callback-Funktion aufgerufen. Es kann nur eine Callback-Funktion für das Ereignis registriert sein. Das Registrieren einer neuen Callback-Funktion führt zum Abmelden der vorhergehenden.

Tabelle 7- 369 RegisterOnConnectionLostCallback() - Native C++

Syntax	`void RegisterOnConnectionLostCallback(`
	`  EventCallback_IRRTM in_CallbackFunction`
	`);`

Parameter	• `EventCallback_IRRTM in_CallbackFunction:` Eine Callback-Funktion, die ein Ereignis abonniert. Siehe EventCallback_IRRTM (Seite 321).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

RegisterOnConnectionLostEvent()

Wenn das Ereignis eintritt, wird das registrierte Event-Objekt in den signalisierten Zustand gesetzt. Es kann nur ein Event-Objekt für das Ereignis registriert sein. Die Registrierung eines neuen Event-Objekts führt zum Löschen des vorhergehenden.

Tabelle 7- 370 RegisterOnConnectionLostEvent() - Native C++

Syntax	`void RegisterOnConnectionLostEvent();`
	`void RegisterOnConnectionLostEvent(`
	`  HANDLE* in_Event`
	`);`

Parameter	• `None:` Ein internes Event-Objekt wird registriert. • `HANDLE* in_Event:` Ein Handle zu einem anwenderspezifischen Event-Objekt. Das Event-Objekt wird registriert.
Rückgabewerte	Keine

Tabelle 7- 371 RegisterOnConnectionLostEvent() - .NET (C#)

Syntax	`void RegisterOnConnectionLostEvent();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnConnectionLostCallback()

Meldet die Callback-Funktion ab. Wenn das Ereignis eintritt, wird keine Callback-Funktion aufgerufen.

Tabelle 7- 372 UnregisterOnConnectionLostCallback() - Native C++

Syntax	`void UnregisterOnConnectionLostCallback();`
Parameter	Keine
Rückgabewerte	Keine

UnregisterOnConnectionLostEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 373 UnregisterOnConnectionLostEvent() - Native C++

Syntax	`void UnregisterOnConnectionLostEvent();`
Parameter	Keine
Rückgabewerte	Keine

Tabelle 7- 374 UnregisterOnConnectionLostEvent() - .NET (C#)

Syntax	`void UnregisterOnConnectionLostEvent();`
Parameter	Keine
Rückgabewerte	Keine

WaitForOnConnectionLostEvent()

Die Funktion blockiert das Programm solange, bis das registrierte Event-Objekt im signalisierten Zustand ist oder bis das Timeout-Intervall überschritten ist.

Tabelle 7- 375 WaitForOnConnectionLostEvent() - Native C++

Syntax	`bool WaitForOnConnectionLostEvent();`
	`bool WaitForOnConnectionLostEvent(`
	`UINT32 in_Time_ms )`
	`;`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UINT32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true:` Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false:` Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

Tabelle 7- 376 WaitForOnConnectionLostEvent() - .NET (C#)

Syntax	`bool WaitForOnConnectionLostEvent();`
	`bool WaitForOnConnectionLostEvent(`
	`UInt32 in_Time_ms )`
	`;`

Parameter	• `None:` Das Zeitlimit ist auf `INFINITE` gesetzt. • `UInt32 in_Time_ms:` Wert für das Zeitlimit in Millisekunden.
Rückgabewerte	• `true`: Wenn das Event-Objekt in den signalisierten Zustand gesetzt wurde. • `false`: Wenn während des definierten Zeitlimits kein Ereignis empfangen wurde.

7.8 Datentypen

Hinweis

Nicht unterstützte Datentypen

Die Runtime API unterstützt nicht die Datentypen `STRING` und `WSTRING`.

Unterstützte Datentypen

In der PLCSIM Advanced V3.0 unterstützt die Runtime API die Datentypen der S7-1500-CPUs.

Datentypen konvertieren

Alle Datentypen werden beim Schreiben nicht BCD-kodiert übergeben, sondern auf primitive Datentypen gemappt.

Die Datentypen Zähler, Datum und Uhrzeit müssen BCD-kodiert an die API übergeben werden, damit die Werte in den Zähler geschrieben und beim Lesen keine falschen Werte zurückgegeben werden.

Führen Sie für diese Datentypen vor dem Schreiben eine BCD-Konvertierung und nach dem Lesen eine BCD-Rückkonvertierung durch.

Beispiel:

Wenn der Wert 999 als 2457_H an die API übergeben wird, dann modifiziert `Write` den Wert 2457_H zu 999. Ohne BCD-Konvertierung gibt es keinen `UInt16`-Wert und `Write` schreibt überhaupt keinen Wert.

Weitere Informationen

Informationen zu Datentypen und zur Konvertierung erhalten Sie im Kapitel "Datentypen" im Systemhandbuch SIMATIC STEP 7 Basic/Professional (https://support.industry.siemens.com/cs/ww/de/view/109755202).

7.8.1 DLL-Importfunktionen (Native C++)

7.8.1.1 ApiEntry_Initialize

Beschreibung

Typ des zentralen Eintrittspunkts für die API-Bibliothek (DLL).

Tabelle 7- 377 ApiEntry_Initialize - Native C++

Syntax	`typedef ERuntimeErrorCode(*ApiEntry_Initialize)( ISimulationRuntimeManager** out_RuntimeManagerInterface );`	
Parameter	• `ISimulationRuntimeManager** out_SimulationRuntimeManagerInterface:` Zeiger auf einen Runtime Manager Schnittstellenzeiger. Der Zeiger muss mit NULL initialisiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt. • `UINT32 in_InterfaceVersion:` Die Version der API-Schnittstelle, die geladen werden soll: `API_DLL_INTERFACE_VERSION`.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WRONG_ARGUMENT`	Der Zeiger auf die Runtime Manager Schnittstelle ist NULL.
	`SREC_WRONG_VERSION`	Die Version der genutzten Schnittstelle passt nicht zur Version der API-Bibliothek (DLL).
	`SREC_CONNECTION_ERROR`	Zum Runtime Manager kann keine Verbindung hergestellt werden.

7.8.1.2 ApiEntry_DestroyInterface

Beschreibung

Typ des Eintrittspunkts für DestroyInterface (Seite 118).

Tabelle 7- 378 ApiEntry_DestroyInterface - Native C++

Syntax	`typedef ERuntimeErrorCode(*ApiEntry_DestroyInterface)( IBaseInterface* in_Interface );`	
Parameter	• `IBaseInterface* in_Interface:` Die Schnittstelle, die gelöscht werden soll.	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WRONG_ARGUMENT`	Der Zeiger auf die Schnittstelle ist NULL.

7.8.2 Event Callback-Funktionen (Native C++)

7.8.2.1 EventCallback_VOID

Beschreibung

Tabelle 7- 379 EventCallback_VOID - Native C++

Syntax	`typedef void (*EventCallback_VOID)();`
Parameter	Keine
Rückgabewerte	Keine

7.8.2.2 EventCallback_SRCC_UINT32_UINT32_INT32

Beschreibung

Tabelle 7- 380 EventCallback_SRCC_UINT32_UINT32_INT32 - Native C++

Syntax	`ERuntimeConfigChanged in_RuntimeConfigChanged,`
	`  UINT32 in_Param1,`
	`  UINT32 in_Param2,`
	`  INT32 in_Param3`
	`);`			

Parameter	`ERuntimeConfigChanged in_RuntimeConfigChanged`	`UInt32 in_Param1`	`UInt32 in_Param2`	`Int32 in_Param3`
	`SRCC_INSTANCE_REGISTERED`	-	-	ID der registrierten Instanz
	`SRCC_INSTANCE_UNREGISTERED`	-	-	ID der nicht registrierten Instanz
	`SRCC_CONNECTION_OPENED`	IP des Remote Runtime Managers	Port des Remote Runtime Managers	-
	`SRCC_CONNECTION_CLOSED`	IP des Remote Runtime Managers	Port des Remote Runtime Managers	-
	`SRCC_PORT_OPENED`	Der offene Port	-	-
	`SRCC_PORT_CLOSED`	-	-	-
Rückgabewerte	Keine			

7.8.2.3 EventCallback_SRRSI_AD

Beschreibung

Tabelle 7- 381 EventCallback_SRRSI_AD - Native C++

Syntax	`typedef void (*EventCallback_SRRSI_AD)(`
	` EAutodiscoverType in_AutodiscoverMsg,`
	` SAutodiscoverData in_AutodiscoverData`
	`);`

Parameter	• `in_AutodiscoverMsg`: Ein Wert aus der Liste der vordefinierten Typen von Ereignissen, siehe EAutodiscoverType (Seite 368). – `SRRSI_DISCOVER_STARTED`, wenn der Identifizierungs-Vorgang durch einen erfolgreichen Aufruf der Funktion `RunAutodisover()` gestartet wurde. – `SRRSI_DISCOVER_DATA`, wenn ein Runtime Manager im Netzwerk durch den Identifizierungs-Vorgang ermittelt werden konnte. Für genaue Informationen zu dem gefundenen Runtime Manager siehe Parameter `in_AutodiscoverData`. – `SRRSI_DISCOVER_FINISHED` wenn der Identifizierungs-Vorgang nach Ablauf der über den Parameter „in_Timeout“ definierten Zeit abgeschlossen wurde. – `SRRSI_DISCOVER_STARTED` und `SRRSI_DISCOVER_FINISHED` werden immer getriggert, auch wenn keine Daten empfangen werden. • `in_AutodiscoverData`: Daten vom Remote Runtime Manager. Der Parameter enthält nur dann gültige Daten, wenn `in_AutodiscoverMsg` = `SRRSI_DISCOVER_DATA`. Ansonsten wird er mit 0 initialisiert. Siehe SAutodiscoverData (Seite 393).
Rückgabewerte	Keine

7.8.2.4 EventCallback_IRRTM

Beschreibung

Tabelle 7- 382 EventCallback_IRRTM - Native C++

Syntax	`typedef void (*EventCallback_IRRTM)(`
	`  IRemoteRuntimeManager* in_Sender`
	`);`

Parameter	• `IRemoteRuntimeManager* in_Sender:` Eine Schnittstelle des Remote Runtime Managers, die dieses Ereignis empfängt.
Rückgabewerte	Keine

7.8.2.5 EventCallback_II_SREC_ST_SROS_SROS

Beschreibung

Tabelle 7- 383 EventCallback_II_SREC_ST_SROS_SROS - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_SROS_SROS)(`
	`  IInstance* in_Sender,`
	`  ERuntimeErrorCode in_ErrorCode,`
	`  SYSTEMTIME in_SystemTime,`
	`  EOperatingState in_PrevState,`
	`  EOperatingState in_OperatingState`
	`);`	

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `SYSTEMTIME in_SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `EOperatingState in_PrevState:` Der Betriebszustand vor dem Wechsel • `EOperatingState in_OperatingState:` Der aktuelle Betriebszustand	
Rückgabewerte	Keine	
Fehlercodes	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WARNING_TRIAL_MODE_ACTIVE`	Keine Lizenz verfügbar. Sie können die Instanz ohne Einschränkung mit der Trial License nutzen. Danach wird die Instanz abgeschaltet.
	`SREC_LICENSE_NOT_FOUND`	Der Testmodus ist abgelaufen.
	`SREC_COMMUNICATION_INTERFACE_NOT_AVAILABLE`	Ein Problem mit der ausgewählten Kommunikationsschnittstelle ist aufgetreten. Überprüfen Sie Ihre Einstellungen.

7.8.2.6 EventCallback_II_SREC_ST_UINT32_INT64_INT64_UINT32

Beschreibung

Tabelle 7- 384 EventCallback_II_SREC_ST_UINT32_INT64_INT64_UINT32 - Native C++

Syntax	`typedef void (*Event-Callback_II_SREC_ST_UINT32_INT64_INT64_UINT32)( IInstance* in_Sender, ERuntimeErrorCode in_ErrorCode, SYSTEMTIME in_SystemTime, UINT32 in_PipId, INT64 in_TimeSinceSameSyncPoint_ns, INT64 in_TimeSinceAnySyncPoint_ns, UINT32 in_SyncPointCount );`
Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `SYSTEMTIME in_SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `UINT32 in_PipId:` Die ID des Teilprozessabbilds (TPA), das dieses Ereignis auslöst. 0 für den Zykluskontrollpunkt (End of cycle). • `INT64 in_TimeSinceSameSyncPoint_ns:` Die virtuelle Zeit (in Nanosekunden) seit der letzte Synchronisationspunkt derselben Teilprozessabbild-ID erreicht wurde. Für die zeitgesteuerten Betriebsarten (Seite 95): Laufzeit seit dem letzten Aufruf der Funktion `StartProcessing()`. • `INT64 in_TimeSinceAnySyncPoint_ns:` Die virtuelle Zeit (in Nanosekunden) seit der letzte Synchronisationspunkt einer beliebigen Teilprozessabbild-ID erreicht wurde. Für die zeitgesteuerten Betriebsarten (Seite 95): Laufzeit seit dem letzten Aufruf der Funktion `StartProcessing()`. • `UINT32 in_SyncPointCount:` Die Anzahl der Synchronisationspunkte seit dem letzten Ereignis. Wenn die Ereignisse schneller ausgelöst als sie empfangen werden, dann werden mehrere Ereignisse zu einem zusammengefasst. In diesem Fall enthält dieser Wert die Anzahl der Zyklen, seit das letzte Ereignis empfangen wurde.
Rückgabewerte	Keine

7.8.2.7 EventCallback_II_SREC_ST

Beschreibung

Tabelle 7- 385 EventCallback_II_SREC_ST - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `SYSTEMTIME in_SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde.
Rückgabewerte	Keine

7.8.2.8 EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32

Beschreibung

Tabelle 7- 386 EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32 - Native C++

Syntax	`typedef void (*Event-`
	`Callback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode, SYSTEMTIME in_SystemTime,`
	` EInstanceConfigChanged in_InstanceConfigChanged,`
	` UINT32 in_Param1,`
	` UINT32 in_Param2,`
	` UINT32 in_Param3,`
	` UINT32 in_Param4`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `SYSTEMTIME in_SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde.

`EInstanceConfigChanged in_InstanceConfigChanged`	`UINT32 in_Param1`	`UINT32 in_Param2`	`UINT32 in_Param3`	`UINT32 in_Param4`
`SRICC_HARDWARE_SOFTWARE_CHANGED`	-	-	-	-
`SRICC_IP_CHANGED`	Die ID der Schnittstelle	Die neue IP	Die neue Subnetzmaske	Das neue Standard-Gateway

Rückgabewerte	Keine

7.8.2.9 EventCallback_II_SREC_ST_SRLT_SRLM

Beschreibung

Tabelle 7- 387 EventCallback_II_SREC_ST_SRLT_SRLM - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_SRLT_SRLM)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime,`
	` ELEDType in_LEDType,`
	` ELEDMode in_LEDMode,`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `SYSTEMTIME in_SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `ELEDType in_LEDType:` Der LED-Typ, der seinen Zustand wechselte. • `ELEDMode in_LEDMode:` Der neue Zustand der LED-Anzeige.
Rückgabewerte	Keine

7.8.2.10 EventCallback_II_SREC_ST_SDRI

Beschreibung

Tabelle 7- 388 EventCallback_II_SREC_ST_SDRI - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_SDRI)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime,`
	` SDataRecordInfo in_DataRecordInfo`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `SYSTEMTIME in_SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `SDataRecordInfo in_DataRecordInfo:` Die Struktur `SDataRecordInfo` enthält folgende Informationen: – Die HW-Kennung, von der die CPU den Datensatz lesen möchte – Den Index des abgeholten Datensatzes – Die maximale Größe des Datensatzes, die das IO-Device übertragen kann
Rückgabewerte	Keine

7.8.2.11 EventCallback_II_SREC_ST_SDRI_BYTE

Beschreibung

Tabelle 7- 389 EventCallback_II_SREC_ST_SDRI_BYTE - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_SDRI_BYTE)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime,`
	` SDataRecordInfo in_DataRecordInfo`
	` const BYTE* in_Data`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `SYSTEMTIME in_SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `SDataRecordInfo in_DataRecordInfo:` Die Struktur `SDataRecordInfo` enthält folgende Informationen: – Die HW-Kennung, an die die CPU den Datensatz schreiben möchte – Den Index des gelieferten Datensatzes – Die Größe des Datensatzes • `const BYTE* in_Data:` Der Datensatz. Dieser Zeiger wird ungültig, nachdem die Callback-Funktion zurückkehrt.
Rückgabewerte	Keine

7.8.2.12 EventCallback_II_SREC_ST_UINT32_UINT32

Beschreibung

Tabelle 7- 390 EventCallback_II_SREC_ST_UINT32_UINT32 - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_UINT32_UINT32)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime,`
	` UINT32 in_HardwareIdentifier),`
	`UINT32 in_SequenceNumber`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `SYSTEMTIME in_SYSTEMTIME:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UINT32 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Diagnoseereignis sendet. • `UINT32 in_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinander folgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1. **Hinweis** In einem realen Hardware-System nutzt der IO-Controller die Sequenznummer, um zu überprüfen, ob er einen Prozessalarm verloren hat. Bei der Simulation stellt die Sequenznummer den Bezug zwischen Alarm-Request und der zugehörigen azyklischen Meldung her.
Rückgabewerte	Keine

7.8.2.13 EventCallback_II_SREC_ST_UINT32_UINT32_EPET_UINT32

Beschreibung

Tabelle 7- 391 EventCallback_II_SREC_ST_UINT32_UINT32_EPET_UINT32 - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_UINT32_UINT32_EPET_UINT32)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime,`
	` UINT32 in_HardwareIdentifier,`
	`UINT32 in_Channel,`
	` EProcessEventType in_ProcessEventType,`
	`UINT32 in_SequenceNumber`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `SYSTEMTIME in_SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UINT32 in_HardwareIdentifier:` Die HW-Kennung des IO-Moduls, das das Prozessereignis sendet. • `UINT32 in_Channel:` Der Kanal des IO-Moduls, das das Prozessereignis sendet. • `EProcessEventType in_ProcessEventType:` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen für S7-Module, siehe EProcessEventType (Seite 390). • `UINT32 in_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinander folgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1. **Hinweis** In einem realen Hardware-System nutzt der IO-Controller die Sequenznummer, um zu überprüfen, ob er einen Prozessalarm verloren hat. Bei der Simulation stellt die Sequenznummer den Bezug zwischen Alarm-Request und der zugehörigen azyklischen Meldung her.
Rückgabewerte	Keine

7.8.2.14 EventCallback_II_SREC_ST_UINT32_EPPET_UINT32

Beschreibung

Tabelle 7- 392 EventCallback_II_SREC_ST_UINT32_EPPET_UINT32 - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_UINT32_EPPET_UINT32)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SYSTEMTIME,`
	` UINT32 in_HardwareIdentifier,`
	`EPullOrPlugEventType in_PullOrPlugEventType,`
	`UINT32 in_SequenceNumber`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `SYSTEMTIME in_SYSTEMTIME:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UINT32 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Ziehen/Stecken-Ereignis sendet. • `EPullOrPlugEventType in_PullOrPlugEventType:` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen für S7-Module, siehe EPullOrPlugEventType (Seite 389). • `UINT32 in_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinander folgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1. **Hinweis** In einem realen Hardware-System nutzt der IO-Controller die Sequenznummer, um zu überprüfen, ob er einen Prozessalarm verloren hat. Bei der Simulation stellt die Sequenznummer den Bezug zwischen Alarm-Request und der zugehörigen azyklischen Meldung her.
Rückgabewerte	Keine

7.8.2.15 EventCallback_II_SREC_ST_UINT32_ERSFET

Beschreibung

Tabelle 7- 393 EventCallback_II_SREC_ST_UINT32_ERSFET - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_UINT32_ERSFET)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime,`
	` UINT32 in_HardwareIdentifier,`
	`ERackOrStationFaultType in_EventType`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `SYSTEMTIME in_SYSTEMTIME:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UINT32 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Diagnoseereignis sendet. • `ERackOrStationFaultType in_EventType:` Ein Wert aus der Liste der vordefinierten RackOrStationFault-Ereignis-Typen. Siehe ERackOrStationFaultType (Seite 392).
Rückgabewerte	Keine

7.8.2.16 EventCallback_II_SREC_ST_UINT32

Beschreibung

Tabelle 7- 394 EventCallback_II_SREC_ST_UINT32 - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_UINT32)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime,`
	` UINT32 in_HardwareIdentifier);`
	`);`

Parameter	• `IInstance* in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `SYSTEMTIME in_SYSTEMTIME:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UINT32 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Status-, Update- oder Profile-Ereignis generiert. Die Kennung muss zu einer Hardware-Komponente im aktuell geladenen Projekt gehören.
Rückgabewerte	Keine

7.8.3 Delegat Definitionen (Managed Code)

7.8.3.1 Delegate_Void

Beschreibung

Tabelle 7- 395 Delegate_Void - .NET (C#)

Syntax	`delegate void Delegate_Void();`
Parameter	Keine
Rückgabewerte	Keine

7.8.3.2 Delegate_SRCC_UINT32_UINT32_INT32

Beschreibung

Tabelle 7- 396 Delegate_SRCC_UINT32_UINT32_INT32 - .NET (C#)

Syntax	`delegate void Delegate_SRCC_UINT32_UINT32_INT32(`
	` ERuntimeConfigChanged in_RuntimeConfigChanged,`
	` UInt32 in_Param1,`
	` UInt32 in_Param2,`
	` Int32 in_Param3`
	`);`			

Parameter	`ERuntimeConfigChanged in_RuntimeConfigChanged`	`UInt32 in_Param1`	`UInt32 in_Param2`	`Int32 in_Param3`
	`InstanceRegistered`	-	-	ID der registrierten Instanz
	`InstanceUnregistered`	-	-	ID der nicht registrierten Instanz
	`ConnectionOpened`	IP des Remote Runtime Managers	Port des Remote Runtime Managers	-
	`ConnectionClosed`	IP des Remote Runtime Managers	Port des Remote Runtime Managers	-
	`PortOpened`	Der offene Port	-	-
	`PortClosed`	-	-	-
Rückgabewerte	Keine			

7.8.3.3 Delegate_SRRSI_AD

Beschreibung

Tabelle 7- 397 Delegate_SRRSI_AD - .NET (C#)

Syntax	`delegate void Delegate_SRRSI_AD(`
	` EAutodiscoverType in_AutodiscoverType,`
	` SAutodiscoverData in_AutodiscoverData`
	`);`

Parameter	• `in_AutodiscoverType` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen, siehe EAutodiscoverType (Seite 393). – `AutodiscoverStarted`, wenn der Identifizierungs-Vorgang durch einen erfolgreichen Aufruf der Funktion `RunAutodisover()` gestartet wurde. – `AutodiscoverData`., wenn ein Runtime Manager im Netzwerk durch den Identifizierungs-Vorgang ermittelt werden konnte. Für genaue Informationen zu dem gefundenen Runtime Manager siehe Parameter `in_AutodiscoverData`. – `AutodiscoverFinished`, wenn der Identifizierungs-Vorgang nach Ablauf der über den Parameter „in_Timeout“ definierten Zeit abgeschlossen wurde. – `AutodiscoverStarted` und `AutodiscoverFinished` werden immer getriggert, auch wenn keine Daten empfangen werden. • `in_AutodiscoverData` Daten vom Remote Runtime Manager. Der Parameter enthält nur dann gültige Daten, wenn `in_AutodiscoverType` = `AutodiscoverData`. Ansonsten wird er mit 0 initialisiert. Siehe SAutodiscoverData (Seite 368).
Rückgabewerte	Keine

7.8.3.4 Delegate_II_EREC_DT

Beschreibung

Tabelle 7- 398 Delegate_II_EREC_DT - .NET (C#)

Syntax	`delegate void Delegate_II_EREC_DT (`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde.
Rückgabewerte	Keine

7.8.3.5 Delegate_II_EREC_DT_EOS_EOS

Beschreibung

Tabelle 7- 399 Delegate_II_EREC_DT_EOS_EOS - .NET (C#)

Syntax	`delegate void Delegate_II_EREC_DT_EOS_EOS(`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` EOperatingState in_PrevState,`
	` EOperatingState in_OperatingState`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `EOperatingState in_PrevState:` Der Betriebszustand vor dem Wechsel. • `EOperatingState in_OperatingState:` Der aktuelle Betriebszustand.

Rückgabewerte	Keine	
Fehlercodes	Runtime Fehlercode	Bedingung
	`ERuntimeErrorCode.OK`	Die Funktion ist erfolgreich.
	`ERuntimeErrorCode.WarningTrial-ModeActive`	Keine Lizenz verfügbar. Sie können die Instanz ohne Einschränkung mit der Trial License nutzen. Danach wird die Instanz abgeschaltet.
	`ERuntimeErrorCode.LicenseNot-Found`	Der Testmodus ist abgelaufen.
	`ERuntimeErrorCode.Communication-InterfaceNotAvailable`	Ein Problem mit der ausgewählten Kommunikationsschnittstelle ist aufgetreten. Überprüfen Sie Ihre Einstellungen.

7.8.3.6 Delegate_II_EREC_DT_ELT_ELM

Beschreibung

Tabelle 7- 400 Delegate_II_EREC_DT_ELT_ELM - .NET (C#)

Syntax	`delegate void Delegate_II_EREC_DT_ELT_ELM(`
	`  IInstance in_Sender,`
	`  ERuntimeErrorCode in_ErrorCode,`
	`  DateTime in_DateTime,`
	`  ELEDType in_LEDType,`
	`  ELEDMode in_LEDMode,`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `ELEDType in_LEDType:` Der LED-Typ, der seinen Zustand wechselte. • `ELEDMode in_LEDMode:` Der neue Zustand der LED-Anzeige.
Rückgabewerte	Keine

7.8.3.7 Delegate_II_EREC_DT_UINT32_INT64_INT64_UINT32

Beschreibung

Tabelle 7- 401 Delegate_II_EREC_DT_UINT32_INT64_INT64_UINT32 - .NET (C#)

Syntax	`delegate void Delegate_II_EREC_DT_UINT32_INT64_INT64_UINT32 (`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` UInt32 in_PipId,`
	` Int64 in_TimeSinceSameSyncPoint_ns,`
	` Int64 in_TimeSinceAnySyncPoint_ns,`
	` UInt32 in_SyncPointCount`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UInt32 in_PipId:` Die ID des Teilprozessabbilds (TPA), das dieses Ereignis auslöst. 0 für den Zykluskontrollpunkt (End of cycle). • `Int64 in_TimeSinceSameSyncPoint_ns:` Die virtuelle Zeit (in Nanosekunden) seit der letzte Synchronisationspunkt derselben Teilprozessabbild-ID erreicht wurde. Oder die Prozesszeit für die zeitgesteuerten Betriebsarten (Seite 95). • `Int64 in_TimeSinceAnySyncPoint_ns:` Die virtuelle Zeit (in Nanosekunden) seit der letzte Synchronisationspunkt einer beliebigen Teilprozessabbild-ID erreicht wurde. Oder die Prozesszeit für die zeitgesteuerten Betriebsarten (Seite 95). • `UInt32 in_SyncPointCount:` Die Anzahl der Synchronisationspunkte seit dem letzten Ereignis. Wenn die Ereignisse schneller ausgelöst als sie empfangen werden, dann werden mehrere Ereignisse zu einem zusammengefasst. In diesem Fall enthält dieser Wert die Anzahl der Zyklen, seit das letzte Ereignis empfangen wurde.
Rückgabewerte	Keine

7.8.3.8 Delegate_IRRTM

Beschreibung

Tabelle 7- 402 Delegate_IRRTM - .NET (C#)

Syntax	`delegate void Delegate_IRRTM(`
	`  IRemoteRuntimeManager in_Sender,`
	`);`

Parameter	• `IRemoteRuntimeManager in_Sender:` Eine Schnittstelle des Remote Runtime Managers, die dieses Ereignis empfängt.
Rückgabewerte	Keine

7.8.3.9 Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32

Beschreibung

Tabelle 7- 403 Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32 - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="5"><code>delegate void Dele-
gate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32(
IInstance in_Sender,
ERuntimeErrorCode in_ErrorCode,
DateTime in_DateTime,
EInstanceConfigChanged in_InstanceConfigChanged,
UInt32 in_Param1,
UInt32 in_Param2,
UInt32 in_Param3,
UInt32 in_Param4
);</code></td></tr>
<tr><td>Parameter</td><td colspan="5">• <code>IInstance in_Sender:</code>
Eine Schnittstelle der Instanz, die dieses Ereignis empfängt.
• <code>ERuntimeErrorCode in_ErrorCode:</code>
Ein möglicher Fehlercode.
• <code>DateTime in_DateTime:</code>
Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde.</td></tr>
<tr><td></td><td><code>EIn-
stanceCon-
figChanged
in_In-
stanceCon-
figChanged</code></td><td><code>UInt32
in_Param1</code></td><td><code>UInt32
in_Param2</code></td><td><code>UInt32
in_Param3</code></td><td><code>UInt32
in_Param4</code></td></tr>
<tr><td></td><td><code>Hard-
wareSoft-
wareChanged</code></td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td></td><td><code>IPChanged</code></td><td>Die ID der Schnittstelle</td><td>Die neue IP</td><td>Die neue Subnetz-maske</td><td>Das neue Standard-Gateway</td></tr>
<tr><td>Rückgabewerte</td><td colspan="5">Keine</td></tr>
</table>

7.8.3.10 Delegate_II_EREC_DT_SDRI

Beschreibung

Tabelle 7- 404 Delegate_II_EREC_DT_SDRI - .NET (C#)

Syntax	`delegate void Delegate_II_EREC_DT_SDRI (`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` SDataRecordInfo in_DataRecordInfo`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `SDataRecordInfo in_DataRecordInfo:` Die Struktur `SDataRecordInfo` enthält folgende Informationen: – Die HW-Kennung, an die die CPU den Datensatz schreiben möchte – Den Index des gelieferten Datensatzes – Die Größe des Datensatzes – Den Datensatz
Rückgabewerte	Keine

7.8.3.11 Delegate_II_EREC_DT_SDR

Beschreibung

Tabelle 7- 405 Delegate_II_EREC_DT_SDR - .NET (C#)

Syntax	`delegate void Delegate_II_EREC_DT_SDR (`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` SDataRecord in_DataRecord`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `SDataRecord in_DataRecord:` Die Struktur `SDataRecord` enthält folgende Informationen: – Die HW-Kennung, an die die CPU den Datensatz schreiben möchte – Den Index des gelieferten Datensatzes – Die Größe des Datensatzes – Den Datensatz
Rückgabewerte	Keine

7.8.3.12 Delegate_SREC_ST_UINT32_EPPET_UINT32

Beschreibung

Tabelle 7- 406 Delegate_SREC_ST_UINT32_EPPET_UINT32 - .NET (C#)

Syntax	`delegate void Delegate_SREC_ST_UINT32 (`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` UInt32 in_HardwareIdentifier,`
	`EPullOrPlugEventType in_PullOrPlugEventType,`
	`UInt32 in_SequenceNumber`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `UInt32 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Ziehen/Stecken-Ereignis sendet. • `EPullOrPlugEventType in_PullOrPlugEventType:` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen für S7-Module, siehe EPullOrPlugEventType (Seite 389). • `UInt32 in_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinander folgende Nummer zu. Entsprechend den PROFINET-Standards ist die Sequenznummer nur 10 Bits weit, sie geht von 1 bis 0x7FF. Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1. **Hinweis** In einem realen Hardware-System nutzt der IO-Controller die Sequenznummer, um zu überprüfen, ob er einen Prozessalarm verloren hat. Bei der Simulation stellt die Sequenznummer den Bezug zwischen Alarm-Request und der zugehörigen azyklischen Meldung her.
Rückgabewerte	Keine

7.8.3.13 Delegate_SREC_ST_UINT32_UINT32_EPET_UINT32

Beschreibung

Tabelle 7- 407 Delegate_SREC_ST_UINT32_UINT32_EPET_UINT32 - Native C++

Syntax	`delegate void Delegate_SREC_ST_UINT32_UINT32_EPET_UINT32(`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` UInt32 in_HardwareIdentifier`
	`UInt32 in_Channel,`
	` EProcessEventType in_ProcessEventType,`
	`UInt32 in_SequenceNumber`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UInt32 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das das Prozessereignis sendet. • `UInt32 in_Channel:` Der Kanal des IO-Moduls, das das Prozessereignis sendet. • `EProcessEventType in_ProcessEventType:` Ein Wert aus der Liste der vordefinierten Typen von Ereignissen für S7-Module, siehe EProcessEventType (Seite 390). • `UInt32 in_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinander folgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1. **Hinweis** In einem realen Hardware-System nutzt der IO-Controller die Sequenznummer, um zu überprüfen, ob er einen Prozessalarm verloren hat. Bei der Simulation stellt die Sequenznummer den Bezug zwischen Alarm-Request und der zugehörigen azyklischen Meldung her.
Rückgabewerte	Keine

7.8.3.14 Delegate_SREC_ST_UINT32

Beschreibung

Tabelle 7- 408 Delegate_SREC_ST_UINT32 - .NET (C#)

Syntax	`delegate void Delegate_SREC_ST_UINT32 (`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` UInt32 in_HardwareIdentifier`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `UInt32 in_HardwareIdentifier:` Die ID des Moduls, das das Status-, Update- oder Profile-Ereignis generiert.
Rückgabewerte	Keine

7.8.3.15 Delegate_SREC_ST_UINT32_UINT32

Beschreibung

Tabelle 7- 409 Delegate_SREC_ST_UINT32_UINT32 - .NET (C#)

Syntax	`delegate void Delegate_SREC_ST_UINT32 (`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` UInt32 in_HardwareIdentifier`
	`UInt32 in_SequenceNumber`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `UInt32 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das einen Diagnoseeintrag sendet. • `UInt32 in_SequenceNumber:` PLCSIM Advanced weist jedem Alarmereignis eine eindeutige aufeinanderfolgende Nummer zu. Nach PROFINET-Standard ist die Sequenznummer 10 Bits weit (1 bis $7FF_H$). Wenn die höchste Nummer erreicht ist, startet die Nummerierung wieder bei 1. **Hinweis** In einem realen Hardware-System nutzt der IO-Controller die Sequenznummer, um zu überprüfen, ob er einen Prozessalarm verloren hat. Bei der Simulation stellt die Sequenznummer den Bezug zwischen Alarm-Request und der zugehörigen azyklischen Meldung her.
Rückgabewerte	Keine

7.8.3.16 Delegate_SREC_ST_UINT32_ERSFET

Beschreibung

Tabelle 7- 410 Delegate_SREC_ST_UINT32_ERSFET - .NET (C#)

Syntax	`delegate void Delegate_SREC_ST_UINT32_ERSFET(`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` UInt32 in_HardwareIdentifier,`
	`ERackOrStationFaultType in_EventType`
	`);`

Parameter	• `IInstance in_Sender:` Eine Schnittstelle der Instanz, die dieses Ereignis empfängt. • `ERuntimeErrorCode in_ErrorCode:` Ein möglicher Fehlercode. • `DateTime in_DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde. • `UInt32 in_HardwareIdentifier:` Die HW-Kennung des Moduls oder Submoduls, das einen Diagnoseeintrag sendet. • `ERackOrStationFaultType in_EventType:` Ein Wert aus der Liste der vordefinierten RackOrStationFault-Ereignis-Typen. Siehe ERackOrStationFaultType (Seite 392).
Rückgabewerte	Keine

7.8.4 Definitionen und Konstanten

Folgende Bezeichner werden in der API verwendet:

Tabelle 7- 411 Definitionen - Native C++

Bezeichner	Wert	Beschreibung
`DINSTANCE_NAME_MAX_LENGTH`	64	Der eindeutige Name einer Instanz muss kleiner sein als dieser Wert.
`DSTORAGE_PATH_MAX_LENGTH`	130	Die maximale Pfadlänge zur virtuellen Memory Card. Einschließlich der NULL-Terminierung.
`DTAG_NAME_MAX_LENGTH`	300	Die maximale Länge des Namens einer PLC-Variablen. Einschließlich der NULL-Terminierung.
`DTAG_ARRAY_DIMENSION`	6	Die maximale Anzahl der Dimension eines multi-dimensionalen Feldes.
`DCONTROLLER_NAME_MAX_LENGTH`	128	Die maximale Länge des Controller-Namens. Einschließlich der NULL-Terminierung.

Bezeichner	Wert	Beschreibung
DCONTROLLER_SHORT_DESIGNATION_MAX_LENGTH	32	Die maximale Länge der Kurzbezeichnung des Controllers (CPU-Typ). Einschließlich der NULL-Terminierung.
DALARM_NOTIFICATION_MAX_DIAG_EVENTS	100	Die maximale Anzahl der Diagnoseereignisse, die in einer Diagnosemeldung gesendet werden.
DPROCESS_EVENT_NAME_MAX_LENGTH	123	Die maximale Länge des Namens für das Prozessereignis. Einschließlich der NULL-Terminierung.
DPROCESS_EVENTS_MAX_ITEMS	256	Die maximale Anzahl konfigurierbaren Prozessereignisse.
DMODULE_STATE_OK	0	AlarmNotification: Modulstatus OK
DMODULE_STATE_ERROR	1	AlarmNotification: Modulstatus fehlerhaft
DMODULE_STATE_MAINT_DEMANDED	2	AlarmNotification: Wartungsanforderung
DMODULE_STATE_MAINT_REQUIRED	4	AlarmNotification: Wartungsbedarf

Tabelle 7- 412 Konstanten - .NET (C#)

Bezeichner	Wert	Beschreibung
RuntimeConstants.InstanceNameLength	64	Der eindeutige Name einer Instanz muss kleiner sein als dieser Wert.
RuntimeConstants.StoragePathMaxLength	130	Die maximale Pfadlänge zur virtuellen Memory Card. Einschließlich der NULL-Terminierung.
RuntimeConstants.TagNameMaxLength	300	Die maximale Länge des Namens einer PLC-Variablen. Einschließlich der NULL-Terminierung.
RuntimeConstants.TagArrayDimension	6	Die maximale Anzahl der Dimension eines multidimensionalen Feldes.
RuntimeConstants.ControllerNameMaxLength	128	Die maximale Länge des Controller-Namens. Einschließlich der NULL-Terminierung.
RuntimeConstants.ControllerShortDesignationMaxLength	32	Die maximale Länge der Kurzbezeichnung des Controllers (CPU-Typ). Einschließlich der NULL-Terminierung.
ModuleState.Ok	0	AlarmNotification: Modulstatus OK
ModuleState.Error	1	AlarmNotification: Modulstatus fehlerhaft
ModuleState. MaintenanceDemanded	2	AlarmNotification: Wartungsanforderung
ModuleState. MaintenanceRequired	4	AlarmNotification: Wartungsbedarf

7.8.5 Unions (Native C++)

7.8.5.1 UIP

Beschreibung

Enthält eine IPv4-Adresse.

Tabelle 7- 413 UIP - Native C++

Syntax	`union UIP`
	`{`
	`  DWORD IP;`
	`  BYTE IPs[4];`
	`};`

Member	• `DWORD IP:` Die IP-Adresse in einem einzelnen DWORD • `BYTE IPs[4]:` Die vier Elemente der IP in absteigender Reihenfolge
Beispiel	Beispiel für eine IP-Adresse: 192.168.0.1 UIP.IP = 0xC0A80001 UIP.IPs[3] = 192, UIP.IPs[2] = 168, UIP.IPs[1] = 0, UIP.IPs[0] = 1

7.8.5.2 UDataValue

Beschreibung

Enthält den Wert einer PLC-Variable.

Tabelle 7- 414 UDataValue - Native C++

Syntax	`union UDataValue`
	`{`
	`bool Bool;`
	`INT8 Int8;`
	`INT16 Int16;`
	`INT32 Int32;`
	`INT64 Int64;`
	`UINT8 UInt8;`
	`UINT16 UInt16;`
	`UINT32 UInt32;`
	`UINT64 UInt64;`
	`float Float;`
	`double Double;`
	`CHAR Char;`
	`WCHAR WChar;`
	`};`

Member	• `bool Bool:` 1-Byte boolscher Wert • `INT8 Int8:` 1-Byte-Ganzzahl mit Vorzeichen • `INT16 Int16:` 2-Byte-Ganzzahl mit Vorzeichen • `INT32 Int32:` 4-Byte-Ganzzahl mit Vorzeichen • `INT64 Int64:` 8-Byte-Ganzzahl mit Vorzeichen • `UINT8 UInt8:` 1-Byte-Ganzzahl ohne Vorzeichen • `UINT16 UInt16:` 2-Byte-Ganzzahl ohne Vorzeichen • `UINT32 UInt32:` 4-Byte-Ganzzahl ohne Vorzeichen • `UINT64 UInt64:` 8-Byte-Ganzzahl ohne Vorzeichen • `float Float:` 4-Byte Gleitkomma Wert • `double Double:` 8-Byte Gleitkomma Wert • `CHAR Char:` 1-Byte Wert Zeichen • `WCHAR WChar:` 2-Byte Wert Zeichen

7.8.6 Strukturen

Folgende Strukturen stehen zur Verfügung:

- SDataValue (Seite 349)
- SDVBNI (Seite 351)
- SDataValueByAddress (Seite 351)
- SDataValueByAddressWithCheck (Seite 352)
- SDataValueByName (Seite 353)
- SDataValueByNameWithCheck (Seite 353)
- SConnectionInfo (Seite 354)
- SInstanceInfo (Seite 354)
- SDimension (Seite 355)
- STagInfo (Seite 356)
- SIP (Seite 358)
- SIPSuite4 (Seite 358)
- SOnSyncPointReachedResult (Seite 360)
- SDataRecordInfo (Seite 362)
- SDataRecord (Seite 363)
- SConfiguredProcessEvents (Seite 363)
- SDiagExtChannelDescription (Seite 365)
- SAutodiscoverData (Seite 368)

7.8.6.1 SDataValue

Beschreibung

Die Struktur enthält den Wert und den Typ einer PLC-Variablen.

Tabelle 7- 415 SDataValue - Native C++

Syntax	`struct SDataValue`
	`{`
	`UDataValue Value;`
	`EPrimitiveDataType Type;`
	`};`

Member	• `UDataValue Value:` Der Wert der PLC-Variablen • `EPrimitiveDataType Type:` Typ der PLC-Variablen

Tabelle 7- 416 SDataValue - .NET (C#)

Syntax	`struct SDataValue`
	`{`
	`bool Bool { get; set; }`
	`Int8 Int8 { get; set; }`
	`Int16 Int16 { get; set; }`
	`Int32 Int32 { get; set; }`
	`Int64 Int64 { get; set; }`
	`UInt8 UInt8 { get; set; }`
	`UInt16 UInt16 { get; set; }`
	`UInt32 UInt32 { get; set; }`
	`UInt64 UInt64 { get; set; }`
	`float Float { get; set; }`
	`double Double { get; set; }`
	`sbyte Char { get; set; }`
	`char WChar { get; set; }`
	`EPrimitiveDataType Type { get; set; }`
	`}`

Member	• `bool Bool:` 1-Byte boolscher Wert • `Int8 Int8:` 1-Byte-Ganzzahl mit Vorzeichen • `Int16 Int16:` 2-Byte-Ganzzahl mit Vorzeichen • `Int32 Int32:` 4-Byte-Ganzzahl mit Vorzeichen • `Int64 Int64:` 8-Byte-Ganzzahl mit Vorzeichen • `UntT8 UInt8:` 1-Byte-Ganzzahl ohne Vorzeichen • `UInt16 UInt16:` 2-Byte-Ganzzahl ohne Vorzeichen • `UInt32 UInt32:` 4-Byte-Ganzzahl ohne Vorzeichen • `UInt64 UInt64:` 8-Byte-Ganzzahl ohne Vorzeichen • `float Float:` 4-Byte Gleitkomma Wert • `double Double:` 8-Byte Gleitkomma Wert • `sbyte Char:` 1-Byte Wert Zeichen • `char WChar:` 2-Byte Wert Zeichen • `EPrimitiveDataType Type:` Typ der PLC-Variablen

7.8.6.2 SDVBNI

Beschreibung

Diese Struktur ist nur für den internen Gebrauch. Ändern Sie diese Struktur nicht.

Tabelle 7- 417 SDVBNI - Native C++

Syntax	`struct SDVBNI`

Tabelle 7- 418 SDVBNI - .NET (C#)

Syntax	`struct SDVBNI`

7.8.6.3 SDataValueByAddress

Beschreibung

Diese Struktur repräsentiert eine PLC-Variable, die über ihre Adresse aufgerufen wird.

Tabelle 7- 419 SDataValueByAddress - Native C++

Syntax	`struct SDataValueByAddress`
	`{`
	` UINT32 Offset;`
	` UINT8 Bit;`
	` SDataValue DataValue;`
	` ERuntimeErrorCode ErrorCode;`
	`};`

Tabelle 7- 420 SDataValueByAddress - .NET (C#)

Syntax	`struct SDataValueByAddress`
	`{`
	` UInt32 Offset;`
	` UInt8 Bit;`
	` SDataValue DataValue;`
	` ERuntimeErrorCode ErrorCode;`
	`}`

7.8.6.4 SDataValueByAddressWithCheck

Beschreibung

Diese Struktur repräsentiert eine PLC-Variable, die über ihre Adresse aufgerufen wird.

Tabelle 7- 421 SDataValueByAddressWithCheck - Native C++

Syntax	`struct SDataValueByAddressWithCheck`
	`{`
	` UINT32 Offset;`
	` UINT8 Bit;`
	` SDataValue DataValue;`
	` ERuntimeErrorCode ErrorCode;`
	` bool ValueHasChanged;`
	`};`

Tabelle 7- 422 SDataValueByAddressWithCheck - .NET (C#)

Syntax	`struct SDataValueByAddressWithCheck`
	`{`
	` UInt32 Offset;`
	` UInt8 Bit;`
	` SDataValue DataValue;`
	` ERuntimeErrorCode ErrorCode;`
	` bool ValueHasChanged;`
	`}`

7.8.6.5 SDataValueByName

Beschreibung

Diese Struktur repräsentiert eine PLC-Variable, die über ihren Namen aufgerufen wird.

Tabelle 7- 423 SDataValueByName - Native C++

Syntax	`struct SDataValueByName`
	`{`
	` WCHAR Name[DTAG_NAME_MAX_LENGTH];`
	` SDataValue DataValue;`
	` ERuntimeErrorCode ErrorCode;`
	` SDVBNI Internal;`
	`};`

Tabelle 7- 424 SDataValueByName - .NET (C#)

Syntax	`struct SDataValueByName`
	`{`
	` String Name;`
	` SDataValue DataValue;`
	` ERuntimeErrorCode ErrorCode;`
	` SDVBNI Internal;`
	`}`

7.8.6.6 SDataValueByNameWithCheck

Beschreibung

Diese Struktur repräsentiert eine PLC-Variable, die über ihren Namen aufgerufen wird.

Tabelle 7- 425 SDataValueByNameWithCheck - Native C++

Syntax	`struct SDataValueByNameWithCheck`
	`{`
	` WCHAR Name[DTAG_NAME_MAX_LENGTH];`
	` SDataValue DataValue;`
	` ERuntimeErrorCode ErrorCode;`
	` SDVBNI Internal;`
	` bool ValueHasChanged;`
	`};`

Tabelle 7- 426 SDataValueByNameWithCheck - .NET (C#)

Syntax	`struct SDataValueByNameWithCheck`
	`{`
	` String Name;`
	` SDataValue DataValue;`
	` ERuntimeErrorCode ErrorCode;`
	` SDVBNI Internal;`
	` bool ValueHasChanged;`
	`}`

7.8.6.7 SConnectionInfo

Beschreibung

Diese Struktur enthält die IP-Adresse und den Port einer TCP/IP-Verbindung.

Tabelle 7- 427 SConnectionInfo - Native C++

Syntax	`struct SConnectionInfo`
	`{`
	` UIP IP;`
	` UINT16 Port;`
	`};`

Tabelle 7- 428 SConnectionInfo - .NET (C#)

Syntax	`struct SConnectionInfo`
	`{`
	` SIP IP;`
	` UInt16 Port;`
	`}`

7.8.6.8 SInstanceInfo

Beschreibung

Diese Struktur enthält eine IPv4-Adresse.

Tabelle 7- 429 SInstanceInfo - Native C++

Syntax	`struct SInstanceInfo`
	`{`
	` INT32 ID;`
	` WCHAR Name[DINSTANCE_NAME_MAX_LENGTH];`
	`};`

Member	• `INT32 ID:` Die ID der Instanz • `WCHAR Name[DINSTANCE_NAME_MAX_LENGTH]:` Der Name der Instanz

Tabelle 7- 430 SInstanceInfo - .NET (C#)

Syntax	`struct SInstanceInfo`
	`{`
	` Int32 ID;`
	` String Name;`
	`}`

Member	• `Int32 ID:` Die ID der Instanz • `String Name:` Der Name der Instanz

7.8.6.9 SDimension

Beschreibung

Diese Struktur enthält Informationen zur Dimension eines Feldes.

Tabelle 7- 431 SDimension - Native C++

Syntax	`struct SDimension`
	`{`
	`  INT32 StartIndex;`
	`  UINT32 Count;`
	`};`

Tabelle 7- 432 SDimension - .NET (C#)

Syntax	`struct SDimension`
	`{`
	`  Int32 StartIndex;`
	`  UInt32 Count;`
	`}`

7.8.6.10 STagInfo

Beschreibung

Diese Struktur enthält Informationen zu einer PLC-Variablen.

Tabelle 7- 433 STagInfo - Native C++

Syntax	`struct STagInfo`
	`{`
	`WCHAR Name[DTAG_NAME_MAX_LENGTH];`
	`EArea Area;`
	`EDataType DataType;`
	`EPrimitiveDataType PrimitiveDataType;`
	`UINT16 Size;`
	`UINT32 Offset;`
	`UINT8 Bit;`
	`UINT8 DimensionCount;`
	`UINT32 Index;`
	`UINT32 ParentIndex;`
	`SDimension Dimension[DTAG_ARRAY_DIMENSION];`
	`};`

Member	• `WCHAR Name[DTAG_NAME_MAX_LENGTH]:` Der Name der Variablen • `EArea Area:` Der CPU-Bereich, in dem sich die Variable befindet. • `EDataType DataType:` Der CPU-Datentyp der Variablen • `EPrimitiveDataType PrimitiveDataType:` Der primitive Datentyp der Variablen • `UINT16 Size:` Die Größe der Variablen in Byte • `UINT32 Offset:` Der Byte-Offset der Variablen, wenn sie sich nicht in einem Datenbaustein befindet. • `UINT8 Bit:` Der Bit-Offset der Variablen, wenn sie sich nicht in einem Datenbaustein befindet. • `UINT8 DimensionCount:` Die Anzahl der Dimensionen des Feldes. 0, wenn es sich bei der Variablen nicht um ein Feld handelt. • `UINT32 Index:` Der Index der Variablen • `UINT32 ParentIndex:` Wenn diese Variable in eine weitere Variable eingebettet ist (wie ein Element einer Struktur), dann zeigt dieser Wert den Index der übergeordneten Variablen. Der Wert ist 0, wenn die Variable keine übergeordnete Variable hat. • `SDimension Dimension[DTAG_ARRAY_DIMENSION]:` Informationen zu jeder Dimension des Feldes

Tabelle 7- 434 STagInfo - .NET (C#)

Syntax	`public struct STagInfo`
	`{`
	`String Name;`
	`EArea Area;`
	`EDataType DataType;`
	`EPrimitiveDataType PrimitiveDataType;`
	`UInt16 Size;`
	`UInt32 Offset;`
	`UInt8 Bit;`
	`UInt32 Index;`
	`UInt32 ParentIndex;`
	`SDimension[] Dimension;`
	`}`

Member	• `String Name:` Der Name der Variablen • `EArea Area:` Der CPU-Bereich, in dem sich die Variable befindet. • `EDataType DataType:` Der CPU-Datentyp der Variablen • `EPrimitiveDataType PrimitiveDataType:` Der primitive Datentyp der Variablen • `UInt16 Size:` Die Größe der Variablen in Byte. • `UInt32 Offset:` Der Byte-Offset der Variablen, wenn sie sich nicht in einem Datenbaustein befindet. • `UInt8 Bit:` Der Bit-Offset der Variablen, wenn sie sich nicht in einem Datenbaustein befindet. • `UInt32 Index:` Der Index der Variablen • `UInt32 ParentIndex:` Wenn diese Variable in eine weitere Variable eingebettet ist (wie ein Element einer Struktur), dann zeigt dieser Wert den Index der übergeordneten Variablen. Der Wert ist 0, wenn die Variable keine übergeordnete Variable hat. • `SDimension[] Dimension:` Informationen zu jeder Dimension des Feldes. Leer, wenn die Variable kein Feld ist.

7.8.6.11 SIP

Beschreibung

Diese Struktur enthält eine IPv4-Adresse.

Tabelle 7- 435 SIP - .NET (C#)

Syntax	`struct SIP`
	`{`
	` byte[] IPArray { get; set; }`
	` UInt32 IPDWord { get; set; }`
	` string IPString { get; set; }`
	`}`

Member	• `UInt32 IPDWord:` Die IP-Adresse in einem einzelnen DWORD • `byte[] IPArray:` Die vier Elemente der IP in absteigender Reihenfolge • `string IPString:` Die IPv4-Adresse als String
Beispiel	Beispiel für eine IP-Adresse: 192.168.0.1 SIP.IPDWord = 0xC0A80001 SIP.IPArray[3] = 192, SIP.IPArray[2] = 168, SIP.IPArray[1] = 0, SIP.IPArray[0] = 1 SIP.IPString = "192.168.0.1"

7.8.6.12 SIPSuite4

Beschreibung

Diese Struktur enthält eine IPv4-Suite.

Tabelle 7- 436 SIPSuite4 - Native C++

Syntax	`struct SIPSuite4`
	`{`
	` UIP IPAddress;`
	` UIP SubnetMask;`
	` UIP DefaultGateway;`
	`};`

Member	• `UIP IPAddress:` Die IP-Adresse • `UIP SubnetMask:` Die Subnetzmaske • `UIP DefaultGateway:` Das Standard-Gateway

Tabelle 7- 437 SIPSuite4 - .NET (C#)

Syntax	`struct SIPSuite4`
	`{`
	`SIP IPAddress;`
	`SIP SubnetMask;`
	`SIP DefaultGateway;`
	`}`

Member	• `SIP IPAddress:` Die IP-Adresse • `SIP SubnetMask:` Die Subnetzmaske • `SIP DefaultGateway:` Das Standard-Gateway

7.8.6.13 SOnSyncPointReachedResult

Beschreibung

Diese Struktur enthält die Ergebnisse des OnSyncPointReached-Ereignisses.

Tabelle 7- 438 SOnSyncPointReachedResult - Native C++

Syntax	`struct SOnSyncPointReachedResult`
	`{`
	` ERuntimeErrorCode ErrorCode;`
	` SYSTEMTIME SystemTime;`
	` UINT32 PipId;`
	` INT64 TimeSinceSameSyncPoint_ns;`
	` INT64 TimeSinceAnySyncPoint_ns;`
	` UINT32 SyncPointCount;`
	`};`

Member	• `ERuntimeErrorCode ErrorCode:` – `SREC_TIMEOUT`, wenn während der definierten Zeitspanne kein Ereignis ausgelöst wurde. – `SREC_WARNING_INVALID_CALL`, wenn zuvor keine Funktion `RegisterOnSyncPointReachedEvent` aufgerufen wurde. Siehe ERuntimeErrorCode (Seite 370). • `SYSTEMTIME SystemTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UINT32 PipId:` Die ID des Teilprozessabbilds (TPA), das dieses Ereignis auslöst. 0 für den Zykluskontrollpunkt (End of cycle). • `INT64 TimeSinceSameSyncPoint_ns:` Die virtuelle Zeit (in Nanosekunden) seit der letzte Synchronisationspunkt derselben Teilprozessabbild-ID erreicht wurde. Für die zeitgesteuerten Betriebsarten (Seite 95): Laufzeit seit dem letzten Aufruf der Funktion `StartProcessing()`. • `INT64 TimeSinceAnySyncPoint_ns:` Die virtuelle Zeit (in Nanosekunden) seit der letzte Synchronisationspunkt einer beliebigen Teilprozessabbild-ID erreicht wurde. Für die zeitgesteuerten Betriebsarten (Seite 95): Laufzeit seit dem letzten Aufruf der Funktion `StartProcessing()`. • `UINT32 SyncPointCount:` Die Anzahl der Synchronisationspunkte seit dem letzten Ereignis. Wenn die Ereignisse schneller ausgelöst als sie empfangen werden, dann werden mehrere Ereignisse zu einem zusammengefasst. In diesem Fall enthält dieser Wert die Anzahl der Zyklen, seit das letzte Ereignis empfangen wurde.

Tabelle 7- 439 SOnSyncPointReachedResult - .NET (C#)

Syntax	`struct SOnSyncPointReachedResult`
	`{`
	` ERuntimeErrorCode ErrorCode;`
	` DateTime SystemTime;`
	` UInt32 PipId;`
	` Int64 TimeSinceSameSyncPoint_ns;`
	` Int64 TimeSinceAnySyncPoint_ns;`
	` UInt32 SyncPointCount;`
	`}`

Member	• `ERuntimeErrorCode ErrorCode:` – `ERuntimeErrorCode.Timeout`, wenn während der definierten Zeitspanne kein Ereignis ausgelöst wurde. – `WarningInvalidCall`, wenn zuvor keine Funktion `RegisterOn-SyncPointReachedEvent` aufgerufen wurde. Siehe ERuntimeErrorCode. • `DateTime DateTime:` Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt, als dieses Ereignis ausgelöst wurde. • `UInt32 PipId:` Die ID des Teilprozessabbilds (TPA), das dieses Ereignis auslöst. 0 für den Zykluskontrollpunkt (End of cycle). • `Int64 TimeSinceSameSyncPoint_ns:` Die virtuelle Zeit (in Nanosekunden) seit der letzte Synchronisationspunkt derselben Teilprozessabbild-ID erreicht wurde. Für die zeitgesteuerten Betriebsarten: Laufzeit seit dem letzten Aufruf der Funktion `StartProcessing()`. • `Int64 TimeSinceAnySyncPoint_ns:` Die virtuelle Zeit (in Nanosekunden) seit der letzte Synchronisationspunkt einer beliebigen Teilprozessabbild-ID erreicht wurde. Für die zeitgesteuerten Betriebsarten: Laufzeit seit dem letzten Aufruf der Funktion `StartProcessing()`. • `UInt32 SyncPointCount:` Die Anzahl der Synchronisationspunkte seit dem letzten Ereignis. Wenn die Ereignisse schneller ausgelöst als sie empfangen werden, dann werden mehrere Ereignisse zu einem zusammengefasst. In diesem Fall enthält dieser Wert die Anzahl der Zyklen, seit das letzte Ereignis empfangen wurde.

7.8.6.14 SDataRecordInfo

Beschreibung

Diese Struktur enthält Lesen/Schreiben Datensatz-Informationen.

Tabelle 7- 440 SDataRecordInfo - Native C++

Syntax	`struct SDataRecordInfo`
	`{`
	`UINT32 HardwareId;`
	`UINT32 RecordIdx;`
	`UINT32 DataSize;`
	`};`

Member	• `UINT32 HardwareId:` Die ID des Hardware-Moduls (Hardware-Identifier) • `UINT32 RecordIdx:` Die Datensatznummer • `UINT32 DataSize:` Die Datensatzgröße

Tabelle 7- 441 SDataRecordInfo - .NET (C#)

Syntax	`struct SDataRecordInfo`
	`{`
	`UInt32 HardwareId;`
	`UInt32 RecordIdx;`
	`UInt32 DataSize;`
	`}`

Member	• `UInt32 ID:` Die ID des Hardware-Moduls • `UInt32 RecordIdx:` Die Datensatznummer • `UInt32 DataSize:` Die Datensatzgröße

7.8.6.15 SDataRecord

Beschreibung

Diese Struktur enthält Lesen/Schreiben Datensatz-Informationen und Datensätze.

Tabelle 7- 442 SDataRecord - .NET (C#)

Syntax	`struct SDataRecord`
	`{`
	`UInt32 HardwareId;`
	`byte[] Data`
	`}`

Member	• `SDataRecordInfo Info:` Die Datensatz-Information, siehe SDataRecordInfo (Seite 362) • `byte[] Data:` Die Feldgröße

7.8.6.16 SConfiguredProcessEvents

Beschreibung

Diese Struktur enthält Informationen über die konfigurierten Prozessereignisse.

Tabelle 7- 443 SConfiguredProcessEvents - Native C++

Syntax	`struct SConfiguredProcessEvents`
	`{`
	`UINT16 HardwareIdentifier;`
	`UINT16 Channel;`
	`EProcessEventType ProcessEventType;`
	`WCHAR Name[DPROCESS_EVENT_NAME_MAX_LENGTH];`
	`};`

Member	• `UINT16 HardwareIdentifier:` Die HW-Kennung • `UINT16 Channel:` Der Kanal des IO-Moduls, das das Prozessereignis generiert. • `EProcessEventType ProcessEventType:` Der Typ des konfigurierten Prozessereignisses • `WCHAR Name[DPROCESS_EVENT_NAME_MAX_LENGTH]:` Der Name des Prozessereignisses

Tabelle 7- 444 SConfiguredProcessEvents - .NET (C#)

Syntax	`public struct SConfiguredProcessEvents`
	`{`
	`ushort HardwareIdentifier;`
	`ushort Channel;`
	`EProcessEventType ProcessEventType;`
	`string Name;`
	`}`

Member	• `ushort HardwareIdentifier:` Die HW-Kennung • `ushort Channel:` Der Kanal des IO-Moduls, das das Prozessereignis generiert. • `EProcessEventType ProcessEventType:` Der Typ des konfigurierten Prozessereignisses • `string Name:` Der Name des Prozessereignisses

7.8.6.17 SDiagExtChannelDescription

Beschreibung

Diese Struktur enthält Lesen/Schreiben Datensatz-Informationen und Datensätze.

Tabelle 7- 445 SDiagExtChannelDescription - Native C++

Syntax	`struct SDiagExtChannelDescription`
	`{`
	`UINT16 ChannelNumber;`
	`UINT16 ErrorType;`
	`UINT16 ExtErrorType;`
	`EDiagSeverity Severity;`
	`EDiagProperty Direction;`
	`};`

Member	• `UINT16 ChannelNumber:` Wenn sich der Alarm auf einen bestimmten Kanal des IO-Device bezieht (z. B. Kurzschluss), dann muss dieser Parameter die Nummer des gestörten Kanals enthalten. Wenn der Alarm von einem Modul oder Submodul generiert wird, dann muss die Nummer des Kanals auf 0x8000 gesetzt werden. • `UINT16 ErrorType:` Der Parameter definiert Fehlertypen nach PROFINET-Standard, siehe Absatz "Fehlertypen". • `EDiagSeverity Severity:` Der Wert des Schweregrads für die Diagnose, siehe EDiagSeverity (Seite 391). • `EDiagProperty Direction:` Der Wert für die Kommend/Gehend-Information, siehe EDiagProperty (Seite 391). • `UINT16 ExtErrorType:` Dieser Parameter bietet die Möglichkeit, mehr Details zum Diagnosealarm zu definieren. Dies ist hilfreich in Kombination mit PDEV-Fehlertypen, die für CPU-interne Module generiert werden. Der Standardwert sollte 0 sein.

Tabelle 7- 446 SDiagExtChannelDescription - .NET (C#)

Syntax	`struct SDiagExtChannelDescription`
	`{`
	`UInt16 ChannelNumber;`
	`UInt16 ErrorType;`
	`UInt16 ExtErrorType;`
	`EDiagSeverity Severity;`
	`EDiagProperty Direction;`
	`};`

Member	• `UInt16 ChannelNumber:` Wenn sich der Alarm auf einen bestimmten Kanal des IO-Device bezieht (z. B. Kurzschluss), dann muss dieser Parameter die Nummer des gestörten Kanals enthalten. Wenn der Alarm von einem Modul oder Submodul generiert wird, dann muss die Nummer des Kanals auf 0x8000 gesetzt werden. • `UInt16 ErrorType:` Der Parameter definiert Fehlertypen nach PROFINET-Standard, siehe Absatz "Fehlertypen". • `EDiagSeverity Severity:` Der Wert des Schweregrads für die Diagnose, siehe EDiagSeverity (Seite 391). • `EDiagProperty Direction:` Der Wert für die Kommend/Gehend-Information, siehe EDiagProperty (Seite 391). • `UInt16 ExtErrorType:` Dieser Parameter bietet die Möglichkeit, mehr Details zum Diagnosealarm zu definieren. Dies ist hilfreich in Kombination mit PDEV-Fehlertypen, die für CPU-interne Module generiert werden. Der Standardwert sollte 0 sein.

Fehlertypen

Die folgende Tabelle enthält wichtige Fehlertypen (`ErrorType`) nach PROFINET-Standard:

Tabelle 7- 447 Fehlertypen nach PROFINET-Standard

Wert	Bedeutung
0x0000	Reserviert / Unbekannter Fehler
0x0001	Kurzschluss
0x0002	Unterspannung
0x0003	Überspannung
0x0004	Überlast
0x0005	Übertemperatur
0x0006	Drahtbruch
0x0007	Obergrenze überschritten
0x0008	Untergrenze unterschritten
0x0009	Fehler

Die folgende Tabelle enthält Fehlertypen `ExtChannelErrorType` für `ChannelErrorType` "Remote mismatch":

Tabelle 7- 448 Fehlertypen ExtChannelErrorType

Wert	Bedeutung	Gebrauch
0x0000	Reserviert	-
0x0001 bis 0x7FFF	Herstellerkennung	Alarm/Diagnose
0x8000	Peer name of station mismatch	Alarm/Diagnose
0x8001	Peer name of port mismatch	Alarm/Diagnose
0x8002	Peer RT_CLASS_3 mismatch	Alarm/Diagnose
0x8003	Peer MAU Type mismatch	Alarm/Diagnose

7.8.6.18 SAutodiscoverData

Beschreibung

Diese Struktur enthält die IP-Adresse, den Port, die Runtime-Version und den Namen des Computers, auf dem sich ein Runtime Manager befindet, der bereit ist, eine Remote-Verbindung herzustellen.

Tabelle 7- 449 SAutodiscoverData - Native C++

Syntax	`public struct SAutodiscoverData`
	`{`
	`  UIP IP;`
	`  UINT16 Port;`
	`  DWORD RuntimeVersion;`
	`  WCHAR ComputerName[MAX_COMPUTERNAME_LENGTH + 1];`
	`};`

Tabelle 7- 450 SAutodiscoverData - .NET (C#)

Syntax	`public struct SAutodiscoverData`
	`{`
	`  public SIP IP;`
	`  public ushort Port;`
	`  public uint RuntimeVersion;`
	`  public string ComputerName;`
	`}`

7.8.7 Aufzählungen

Folgende Aufzählungen stehen zur Verfügung:

- ERuntimeErrorCode (Seite 370)
- EArea (Seite 372)
- EOperatingState (Seite 373)
- EOperatingMode (Seite 374)
- ECPUType (Seite 375)
- ECommunicationInterface (Seite 377)
- ELEDType (Seite 378)
- ELEDMode (Seite 379)
- EPrimitiveDataType (Seite 380)
- EDataType (Seite 383)
- ETagListDetails (Seite 387)
- ERuntimeConfigChanged (Seite 388)
- EInstanceConfigChanged (Seite 388)
- EPullOrPlugEventType (Seite 389)
- EProcessEventType (Seite 390)
- EDirection (Seite 390)
- EDiagProperty (Seite 391)
- EDiagSeverity (Seite 391)
- ERackOrStationFaultType (Seite 392)
- ECycleTimeMonitoringMode (Seite 392)
- EAutodiscoverType (Seite 393)

Siehe auch

Globale Funktionen (Native C++) (Seite 122)

7.8.7.1 ERuntimeErrorCode

Beschreibung

Diese Aufzählung enthält alle Fehlercodes, die von der Simulation Runtime API genutzt werden. Die meisten der API-Funktionen liefern einen dieser Fehlercodes zurück. Wenn die Funktionen erfolgreich sind, ist der Rückgabewert immer `SREC_OK` / `OK`. Fehler werden mit negativen Werten zurückgegeben, Warnungen mit positiven.

Tabelle 7- 451 ERuntimeErrorCode - Native C++

Syntax	`enum ERuntimeErrorCode`
	`{`
	`SREC_OK = 0,`
	`SREC_INVALID_ERROR_CODE = -1,`
	`SREC_NOT_IMPLEMENTED = -2,`
	`SREC_INDEX_OUT_OF_RANGE = -3,`
	`SREC_DOES_NOT_EXIST = -4,`
	`SREC_ALREADY_EXISTS = -5,`
	`SREC_UNKNOWN_MESSAGE_TYPE = -6,`
	`SREC_INVALID_MESSAGE_ID = -7,`
	`SREC_WRONG_ARGUMENT = -8,`
	`SREC_WRONG_PIPE = -9,`
	`SREC_CONNECTION_ERROR = -10,`
	`SREC_TIMEOUT = -11,`
	`SREC_MESSAGE_CORRUPT = -12,`
	`SREC_WRONG_VERSION = -13,`
	`SREC_INSTANCE_NOT_RUNNING = -14,`
	`SREC_INTERFACE_REMOVED = -15,`
	`SREC_SHARED_MEMORY_NOT_INITIALIZED = -16,`
	`SREC_API_NOT_INITIALIZED = -17,`
	`SREC_WARNING_ALREADY_EXISTS = 18,`
	`SREC_NOT_SUPPORTED = -19,`
	`SREC_WARNING_INVALID_CALL = 20,`
	`SREC_ERROR_LOADING_DLL = -21,`
	`SREC_SIGNAL_NAME_DOES_NOT_EXIST = -22,`
	`SREC_SIGNAL_TYPE_MISMATCH = -23,`
	`SREC_SIGNAL_CONFIGURATION_ERROR = -24,`
	`SREC_NO_SIGNAL_CONFIGURATION_LOADED = -25,`
	`SREC_CONFIGURED_CONNECTION_NOT_FOUND = -26,`
	`SREC_CONFIGURED_DEVICE_NOT_FOUND = -27,`
	`SREC_INVALID_CONFIGURATION = -28,`
	`SREC_TYPE_MISMATCH = -29,`
	`SREC_LICENSE_NOT_FOUND = -30,`
	`SREC_NO_LICENSE_AVAILABLE = -31,`
	`SREC_WRONG_COMMUNICATION_INTERFACE = -32,`
	`SREC_LIMIT_REACHED = -33,`
	`SREC_NO_STORAGE_PATH_SET = -34,`
	`SREC_STORAGE_PATH_ALREADY_IN_USE = -35,`
	`SREC_MESSAGE_INCOMPLETE = -36,`
	`SREC_ARCHIVE_STORAGE_NOT_CREATED = -37,`
	`SREC_RETRIEVE_STORAGE_FAILURE = -38`
	`SREC_INVALID_OPERATING_STATE = -39,`
	`SREC_INVALID_ARCHIVE_PATH = -40,`
	`SREC_DELETE_EXISTING_STORAGE_FAILED = -41,`
	`SREC_CREATE_DIRECTORIES_FAILED = -42,`
	`SREC_NOT_ENOUGH_MEMORY = -43,`
	`SREC_WARNING_TRIAL_MODE_ACTIVE = 44,`
	`SREC_NOT_RUNNING = -45,`
	`SREC_NOT_EMPTY = -46,`
	`SREC_NOT_UP_TO_DATE = -47,`
	`SREC_COMMUNICATION_INTERFACE_NOT_AVAILABLE = -48,`
	`SREC_WARNING_NOT_COMPLETE = 49,`
	`SREC_VIRTUAL_SWITCH_MISCONFIGURED = -50,`
	`SREC_RUNTIME_NOT_AVAILABLE = -51,`
	`SREC_IS_EMPTY = -52,`
	`SREC_WRONG_MODULE_STATE = -53,`
	`SREC_WRONG_MODULE_TYPE = -54,`
	`SREC_NOT_SUPPORTED_BY_MODULE = -55,`
	`SREC_INTERNAL_ERROR = -56,`
	`SREC_STORAGE_TRANSFER_ERROR = -57,`
	`SREC_ANOTHER_VARIANT_OF_PLCSIM_RUNNING = -58,`
	`SREC_ACCESS_DENIED = -59,`
	`SREC_NOT_ALLOWED_DURING_DOWNLOAD = -60`
	`SREC_AUTODISCOVER_ALREADY_RUNNING = -61,`
	`SREC_INVALID_STORAGE = -62`
	`SREC_WARNING_UNSUPPORTED_PCAP_DRIVER = 63,`
	`SREC_WARNING_RUNNING_ON_TIA_PORTAL_TEST_SUITE = 64`
	`};`

Tabelle 7- 452 ERuntimeErrorCode - .NET (C#)

Syntax	enum ERuntimeErrorCode
	{
	OK = 0,
	InvalidErrorCode = -1,
	NotImplemented = -2,
	IndexOutOfRange = -3,
	DoesNotExist = -4,
	AlreadyExists = -5,
	UnknownMessageType = -6,
	InvalidMessageId = -7,
	WrongArgument = -8,
	WrongPipe = -9,
	ConnectionError = -10,
	Timeout = -11,
	MessageCorrupt = -12,
	WrongVersion = -13,
	InstanceNotRunning = -14,
	InterfaceRemoved = -15,
	SharedMemoryNotInitialized = -16,
	ApiNotInitialized = -17,
	WarningAlreadyExists = 18,
	NotSupported = -19,
	WarningInvalidCall = 20,
	ErrorLoadingDll = -21,
	SignalNameDoesNotExist = -22,
	SignalTypeMismatch = -23,
	SignalConfigurationError = -24,
	NoSignalConfigurationLoaded = -25,
	ConfiguredConnectionNotFound = -26,
	ConfiguredDeviceNotFound = -27,
	InvalidConfiguration = -28,
	TypeMismatch = -29,
	LicenseNotFound = -30,
	NoLicenseAvailable = -31,
	WrongCommunicationInterface = -32,
	LimitReached = -33,
	NoStartupPathSet = -34,
	StartupPathAlreadyInUse = -35,
	MessageIncomplete = -36,
	ArchiveStorageNotCreated = -37,
	RetrieveStorageFailure = -38,
	InvalidOperatingState = -39,
	InvalidArchivePath = -40,
	DeleteExistingStorageFailed = -41,
	CreateDirectoriesFailed = -42,
	NotEnoughMemory = -43,
	WarningTrialModeActive = 44,
	NotRunning = -45,
	NotEmpty = -46,
	NotUpToDate = -47,
	CommunicationInterfaceNotAvailable = -48,
	WarningNotComplete = 49,
	RuntimeNotAvailable = -51,
	IsEmpty = -52,
	WrongModuleState = -53,
	WrongModuleType = -54,
	NotSupportedByModule = -55,
	InternalError = -56,
	StorageTransferError = -57,
	AnotherVariantOfPlcsimRunning = -58,
	AccessDenied = -59,
	NotAllowedDuringDownload = -60,
	AutodiscoverAlreadyRunning = -61,
	InvalidStorage = -62,
	WarningUnsupportedPcapDriver = 63,
	WarningRunningOnTiaPortalTestSuite = 64
	}

7.8.7.2 EArea

Beschreibung

Diese Aufzählung enthält alle PLC-Areas, die verfügbare PLC-Variablen beinhalten.

Tabelle 7- 453 EArea - Native C++

Syntax	`enum EArea`
	`{`
	` SRA_INVALID_AREA = 0,`
	` SRA_INPUT = 1,`
	` SRA_MARKER = 2,`
	` SRA_OUTPUT = 3,`
	` SRA_COUNTER = 4,`
	` SRA_TIMER = 5,`
	` SRA_DATABLOCK = 6,`
	
	` SRA_ENUMERATION_SIZE = 7`
	`};`

Tabelle 7- 454 EArea - .NET (C#)

Syntax	`public enum EArea`
	`{`
	` InvalidArea = 0,`
	` Input = 1,`
	` Marker = 2,`
	` Output = 3,`
	` Counter = 4,`
	` Timer = 5,`
	` DataBlock = 6,`
	`}`

7.8.7.3 EOperatingState

Beschreibung

Diese Aufzählung enthält alle Betriebszustände eines virtuellen Controllers.

Tabelle 7- 455 EOperatingState - Native C++

Syntax	`enum EOperatingState`
	`{`
	` SROS_INVALID_OPERATING_STATE = 0,`
	` SROS_OFF = 1,`
	` SROS_BOOTING = 2,`
	` SROS_STOP = 3,`
	` SROS_STARTUP = 4,`
	` SROS_RUN = 5,`
	` SROS_FREEZE = 6,`
	` SROS_SHUTTING_DOWN = 7,`
	` SROS_HOLD = 8,`
	
	` SROS_ENUMERATION_SIZE = 9`
	`};`

Tabelle 7- 456 EOperatingState - .NET (C#)

Syntax	`enum EOperatingState`
	`{`
	` InvalidOperatingState = 0,`
	` Off = 1,`
	` Booting = 2,`
	` Stop = 3,`
	` Startup = 4,`
	` Run = 5,`
	` Freeze = 6,`
	` ShuttingDown = 7,`
	` Hold = 8`
	`}`

7.8.7.4 EOperatingMode

Beschreibung

Diese Aufzählung enthält alle Betriebsarten eines virtuellen Controllers.

Tabelle 7- 457 EOperatingMode - Native C++

Syntax	`enum EOperatingMode`
	`{`
	` SROM_DEFAULT = 0,`
	` SROM_SINGLE_STEP_C = 1,`
	` SROM_SINGLE_STEP_CT = 2,`
	` SROM_TIMESPAN_SYNCHNRONIZED_C = 3,`
	` SROM_SINGLE_STEP_P = 4,`
	` SROM_TIMESPAN_SYNCHNRONIZED_P = 5,`
	` SROM_SINGLE_STEP_CP = 6,`
	` SROM_SINGLE_STEP_CPT = 7,`
	` SROM_TIMESPAN_SYNCHNRONIZED_CP = 8`
	`};`

Tabelle 7- 458 EOperatingMode - .NET (C#)

Syntax	`public enum EOperatingMode`
	`{`
	` Default = 0,`
	` SingleStep_C = 1,`
	` SingleStep_CT = 2,`
	` TimespanSynchronized_C = 3,`
	` SingleStep_P = 4,`
	` TimespanSynchronized_P = 5,`
	` SingleStep_CP = 6,`
	` SingleStep_CPT = 7,`
	` TimespanSynchronized_CP = 8`
	`}`

7.8.7.5 ECPUType

Beschreibung

Diese Aufzählung enthält alle CPU-Typen, die auf einen virtuellen Controller geladen werden können.

Tabelle 7- 459 ECPUType - Native C++

Syntax	enum ECPUType
	{
	SRCT_1500_Unspecified = 0x000005DC,
	SRCT_1511 = 0x000005E7,
	SRCT_1511v2 = 0x010005E7,
	SRCT_1511v3 = 0x020005E7,
	SRCT_1513 = 0x000005E9,
	SRCT_1513v2 = 0x010005E9,
	SRCT_1513v3 = 0x020005E9
	SRCT_1515 = 0x000005EB,
	SRCT_1515v2 = 0x010005EB,
	SRCT_1516 = 0x000005EC,
	SRCT_1516v2 = 0x010005EC,
	SRCT_1517 = 0x000005ED,
	SRCT_1518 = 0x000005EE,
	SRCT_1511C = 0x000405E7,
	SRCT_1511Cv2 = 0x010405E7,
	SRCT_1512C = 0x000405E8,
	SRCT_1512Cv2 = 0x010405E8,
	SRCT_1511F = 0x000105E7,
	SRCT_1511Fv2 = 0x010105E7,
	SRCT_1511Fv3 = 0x020105E7,
	SRCT_1513F = 0x000105E9,
	SRCT_1513Fv2 = 0x010105E9,
	SRCT_1513Fv3 = 0x020105E9,
	SRCT_1515F = 0x000105EB,
	SRCT_1515Fv2 = 0x010105EB,
	SRCT_1516F = 0x000105EC,
	SRCT_1516Fv2 = 0x010105EC,
	SRCT_1517F = 0x000105ED,
	SRCT_1518F = 0x000105EE,
	SRCT_1511T = 0x000805E7,
	SRCT_1515T = 0x000805EB,
	SRCT_1516T = 0x000805EC,
	SRCT_1517T = 0x000805ED,
	SRCT_1511TF = 0x000905E7,
	SRCT_1515TF = 0x000905EB,
	SRCT_1516TF = 0x000905EC,
	SRCT_1517TF = 0x000905ED,
	SRCT_1518ODK = 0x001005EE,
	SRCT_1518FODK = 0x001105EE,
	SRCT_1518MFP = 0x004005EE,
	SRCT_1518FMFP = 0x004105EE,
	SRCT_ET200SP_Unspecified = 0x000205DC,
	SRCT_1510SP = 0x000205E6,
	SRCT_1510SPv2 = 0x010205E6,
	SRCT_1512SP = 0x000205E8,
	SRCT_1512SPv2 = 0x010205E8,
	SRCT_1510SPF = 0x000305E6,
	SRCT_1510SPFv2 = 0x010305E6,
	SRCT_1512SPF = 0x000305E8,
	SRCT_1512SPFv2 = 0x010305E8
	}

Tabelle 7- 460 ECPUType - .NET (C#)

Syntax	enum ECPUType
	{
	CPU1500_Unspecified = 0x000005DC,
	CPU1511 = 0x000005E7,
	CPU1511v2 = 0x010005E7,
	CPU1513 = 0x000005E9,
	CPU1513v2 = 0x010005E9,
	CPU1511v3 = 0x020005E7,
	CPU1515 = 0x000005EB,
	CPU1515v2 = 0x010005EB,
	CPU1513v3 = 0x020005E9,
	CPU1516 = 0x000005EC,
	CPU1516v2 = 0x010005EC,
	CPU1517 = 0x000005ED,
	CPU1518 = 0x000005EE,
	CPU1511C = 0x000405E7,
	CPU1511Cv2 = 0x010405E7,
	CPU1512C = 0x000405E8,
	CPU1512Cv2 = 0x010405E8,
	CPU1511F = 0x000105E7,
	CPU1511Fv2 = 0x010105E7,
	CPU1511Fv3 = 0x020105E7,
	CPU1513F = 0x000105E9,
	CPU1513Fv2 = 0x010105E9,
	CPU1513Fv3 = 0x020105E9,
	CPU1515F = 0x000105EB,
	CPU1515Fv2 = 0x010105EB,
	CPU1516F = 0x000105EC,
	CPU1516Fv2 = 0x010105EC,
	CPU1517F = 0x000105ED,
	CPU1518F = 0x000105EE,
	CPU1511T = 0x000805E7,
	CPU1515T = 0x000805EB,
	CPU1516T = 0x000805EC,
	CPU1517T = 0x000805ED,
	CPU1511TF = 0x000905E7,
	CPU1515TF = 0x000905EB,
	CPU1516TF = 0x000905EC,
	CPU1517TF = 0x000905ED,
	CPU1518ODK = 0x001005EE,
	CPU1518FODK = 0x001105EE,
	CPU1518MFP = 0x004005EE,
	CPU1518FMFP = 0x004105EE,
	CPUET200SP_Unspecified = 0x000205DC,
	CPU1510SP = 0x000205E6,
	CPU1510SPv2 = 0x010205E6,
	CPU1512SP = 0x000205E8,
	CPU1512SPv2 = 0x010205E8,
	CPU1510SPF = 0x000305E6,
	CPU1510SPFv2 = 0x010305E6,
	CPU1512SPF = 0x000305E8,
	CPU1512SPFv2 = 0x010305E8
	}

7.8.7.6 ECommunicationInterface

Beschreibung

Diese Aufzählung enthält die verfügbaren Kommunikationsschnittstellen eines virtuellen Controllers.

Tabelle 7- 461 ECommunicationInterface - Native C++

Syntax	`enum ECommunicationInterface`
	`{`
	` SRCI_NONE = 0,`
	` SRCI_SOFTBUS = 1,`
	` SRCI_TCPIP = 2,`
	` SRCI_ENUMERATION_SIZE = 3`
	`};`

Tabelle 7- 462 ECommunicationInterface - .NET (C#)

Syntax	`enum ECommunicationInterface`
	`{`
	` None = 0,`
	` Softbus = 1,`
	` TCPIP = 2,`
	`}`

7.8.7.7 ELEDType

Beschreibung

Diese Aufzählung enthält alle LED-Typen eines virtuellen Controllers.

Tabelle 7- 463 ELEDType - Native C++

Syntax	`enum ELEDType`
	`{`
	`SRLT_STOP = 0,`
	`SRLT_RUN = 1,`
	`SRLT_ERROR = 2,`
	`SRLT_MAINT = 3,`
	`SRLT_REDUND = 4,`
	`SRLT_FORCE = 5,`
	`SRLT_BUSF1 = 6,`
	`SRLT_BUSF2 = 7,`
	`SRLT_BUSF3 = 8,`
	`SRLT_BUSF4 = 9,`
	
	`SRLT_ENUMERATION_SIZE = 10`
	`};`

Tabelle 7- 464 ELEDType - .NET (C#)

Syntax	`enum ELEDType`
	`{`
	`Stop = 0,`
	`Run = 1,`
	`Error = 2`
	`Maint = 3,`
	`Redund = 4,`
	`Force = 5,`
	`Busf1 = 6,`
	`Busf2 = 7,`
	`Busf3 = 8,`
	`Busf4 = 9`
	`}`

7.8.7.8 ELEDMode

Beschreibung

Diese Aufzählung enthält alle LED-Zustände eines virtuellen Controllers.

Tabelle 7- 465 ELEDMode - Native C++

Syntax	`enum ELEDMode`
	`{`
	`SRLM_OFF = 0,`
	`SRLM_ON = 1,`
	`SRLM_FLASH_FAST = 2,`
	`SRLM_FLASH_SLOW = 3,`
	`SRLM_INVALID = 4`
	`};`

Tabelle 7- 466 ELEDMode - .NET (C#)

Syntax	`enum ELEDMode`
	`{`
	`Off = 0,`
	`On = 1,`
	`FlashFast = 2,`
	`FlashSlow = 3,`
	`Invalid = 4`
	`}`

7.8.7.9 EPrimitiveDataType

Beschreibung

Diese Aufzählung enthält alle primitiven Datentypen, die von den I/O-Zugriffs-Funktionen genutzt werden.

Tabelle 7- 467 EPrimitiveDataType - Native C++

Syntax	`enum EPrimitiveDataType`
	`{`
	` SRPDT_UNSPECIFIC = 0,`
	` SRPDT_STRUCT = 1,`
	` SRPDT_BOOL = 2,`
	` SRPDT_INT8 = 3,`
	` SRPDT_INT16 = 4,`
	` SRPDT_INT32 = 5,`
	` SRPDT_INT64 = 6,`
	` SRPDT_UINT8 = 7,`
	` SRPDT_UINT16 = 8,`
	` SRPDT_UINT32 = 9,`
	` SRPDT_UINT64 = 10,`
	` SRPDT_FLOAT = 11,`
	` SRPDT_DOUBLE = 12,`
	` SRPDT_CHAR = 13,`
	` SRPDT_WCHAR = 14`
	`};`

Tabelle 7- 468 EPrimitiveDataType - .NET (C#)

Syntax	`enum EPrimitiveDataType`
	`{`
	` Unspecific = 0,`
	` Struct = 1,`
	` Bool = 2,`
	` Int8 = 3,`
	` Int16 = 4,`
	` Int32 = 5,`
	` Int64 = 6,`
	` UInt8 = 7,`
	` UInt16 = 8,`
	` UInt32 = 9,`
	` UInt64 = 10,`
	` Float = 11,`
	` Double = 12,`
	` Char = 13,`
	` WChar = 14`
	`}`

Kompatible primitive Datentypen

Die folgenden Tabellen zeigen die primitiven Datentypen der Anwenderschnittstelle (API) und die Datentypen der PLCSIM Advanced Instanz, die in der gespeicherten Variablentabelle konfiguriert sind. Die Datentypen, die kompatibel verwendbar sind, sind mit "X" markiert.

Tabelle 7- 469 Kompatible primitive Datentypen - Lesen

API	**PLCSIM Advanced Instanz**												
	Bool	INT				UINT				Float	Dou-ble	Char	WChar
		8	16	32	64	8	16	32	64				
Bool	X												
INT8		X											
INT16		X	X			X							
INT32		X	X	X		X	X						
INT64		X	X	X	X	X	X	X					
UINT8						X							
UINT16						X	X						
UINT32						X	X	X					
UINT64						X	X	X	X				
Float										X			
Double											X		
Char												X	
WChar													X

Tabelle 7- 470 Kompatible primitive Datentypen - Schreiben

API	**PLCSIM Advanced Instanz**											
	Bool	INT				UINT				Float	Dou-ble	Char	WChar
		8	16	32	64	8	16	32	64				
Bool	X												
INT8		X	X	X	X								
INT16			X	X	X								
INT32				X	X								
INT64					X								
UINT8			X	X	X	X	X	X	X				
UINT16				X	X		X	X	X				
UINT32					X			X	X				
UINT64									X				
Float										X			
Double											X		
Char												X	
WChar													X

7.8.7.10 EDataType

Beschreibung

Diese Aufzählung enthält alle PLC-Datentypen (STEP 7).

Tabelle 7- 471 EDataType - Native C++

Syntax	`enum EDataType`
	`{`
	` SRDT_UNKNOWN = 0,`
	` SRDT_BOOL = 1,`
	` SRDT_BYTE = 2,`
	` SRDT_CHAR = 3,`
	` SRDT_WORD = 4,`
	` SRDT_INT = 5,`
	` SRDT_DWORD = 6,`
	` SRDT_DINT = 7,`
	` SRDT_REAL = 8,`
	` SRDT_DATE = 9,`
	` SRDT_TIME_OF_DAY = 10,`
	` SRDT_TIME = 11,`
	` SRDT_S5TIME = 12,`
	` SRDT_DATE_AND_TIME = 14,`
	` SRDT_STRUCT = 17,`
	` SRDT_STRING = 19,`
	` SRDT_COUNTER = 28,`
	` SRDT_TIMER = 29,`
	` SRDT_IEC_Counter = 30,`
	` SRDT_IEC_Timer = 31,`
	` SRDT_LREAL = 48,`
	` SRDT_ULINT = 49,`
	` SRDT_LINT = 50,`
	` SRDT_LWORD = 51,`
	` SRDT_USINT = 52,`
	` SRDT_UINT = 53,`
	` SRDT_UDINT = 54,`
	` SRDT_SINT = 55,`
	` SRDT_WCHAR = 61,`
	` SRDT_WSTRING = 62,`
	` SRDT_LTIME = 64,`
	` SRDT_LTIME_OF_DAY = 65,`
	` SRDT_LDT = 66,`

```
SRDT_DTL = 67,
 SRDT_IEC_LTimer = 68,
 SRDT_IEC_SCounter = 69,
 SRDT_IEC_DCounter = 70,
 SRDT_IEC_LCounter = 71,
 SRDT_IEC_UCounter = 72,
 SRDT_IEC_USCounter = 73,
 SRDT_IEC_UDCounter = 74,
 SRDT_IEC_ULCounter = 75,
 SRDT_ERROR_STRUCT = 97,
 SRDT_NREF = 98,
 SRDT_CREF = 101,
 SRDT_AOM_IDENT = 128,
 SRDT_EVENT_ANY = 129,
 SRDT_EVENT_ATT = 130,
 SRDT_EVENT_HWINT = 131,
 SRDT_HW_ANY = 144,
 SRDT_HW_IOSYSTEM = 145,
 SRDT_HW_DPMASTER = 146,
 SRDT_HW_DEVICE = 147,
 SRDT_HW_DPSLAVE = 148,
 SRDT_HW_IO = 149,
 SRDT_HW_MODULE = 150,
 SRDT_HW_SUBMODULE = 151,
 SRDT_HW_HSC = 152,
 SRDT_HW_PWM = 153,
 SRDT_HW_PTO = 154,
 SRDT_HW_INTERFACE = 155,
 SRDT_HW_IEPORT = 156,
 SRDT_OB_ANY = 160,
 SRDT_OB_DELAY = 161,
 SRDT_OB_TOD = 162,
 SRDT_OB_CYCLIC = 163,
 SRDT_OB_ATT = 164,
 SRDT_CONN_ANY = 168,
 SRDT_CONN_PRG = 169,
 SRDT_CONN_OUC = 170,
 SRDT_CONN_R_ID = 171,
 SRDT_PORT = 173,
 SRDT_RTM = 174,
 SRDT_PIP = 175,
 SRDT_OB_PCYCLE = 192,
 SRDT_OB_HWINT = 193,
 SRDT_OB_DIAG = 195,
 SRDT_OB_TIMEERROR = 196,
 SRDT_OB_STARTUP = 197,
 SRDT_DB_ANY = 208,
 SRDT_DB_WWW = 209,
 SRDT_DB_DYN = 210,
 SRDT_DB = 257
};
```

Tabelle 7- 472 EDataType - .NET (C#)

Syntax	`public enum EDataType`
	`{`
	` Unknown = 0,`
	` Bool = 1,`
	` Byte = 2,`
	` Char = 3,`
	` Word = 4,`
	` Int = 5,`
	` DWord = 6,`
	` DInt = 7,`
	` Real = 8,`
	` Date = 9,`
	` TimeOfDay = 10,`
	` Time = 11,`
	` S5Time = 12,`
	` DateAndTime = 14,`
	` Struct = 17,`
	` String = 19,`
	` Counter = 28,`
	` Timer = 29,`
	` IEC_Counter = 30,`
	` IEC_Timer = 31,`
	` LReal = 48,`
	` ULInt = 49,`
	` LInt = 50,`
	` LWord = 51,`
	` USInt = 52,`
	` UInt = 53,`
	` UDInt = 54,`
	` SInt = 55,`
	` WChar = 61,`
	` WString = 62,`
	` LTime = 64,`

	LTimeOfDay = 65, LDT = 66, DTL = 67, IEC_LTimer = 68, IEC_SCounter = 69, IEC_DCounter = 70, IEC_LCounter = 71, IEC_UCounter = 72, IEC_USCounter = 73, IEC_UDCounte = 74, IEC_ULCounter = 75, ErrorStruct = 97, NREF = 98, CREF = 101, Aom_Ident = 128, Event_Any = 129, Event_Att = 130, Event_HwInt = 131, Hw_Any = 144, Hw_IoSystem = 145, Hw_DpMaster = 146, Hw_Device = 147, Hw_DpSlave = 148, Hw_Io = 149, Hw_Module = 150, Hw_SubModule = 151, Hw_Hsc = 152, Hw_Pwm = 153, Hw_Pto = 154, Hw_Interface = 155, Hw_IEPort = 156, OB_Any = 160, OB_Delay = 161, OB_Tod = 162, OB_Cyclic = 163, OB_Att = 164, Conn_Any = 168, Conn_Prg = 169, Conn_Ouc = 170, Conn_R_ID = 171, Port = 173, Rtm = 174, Pip = 175, OB_PCycle = 192, OB_HwInt = 193, OB_Diag = 195, OB_TimeError = 196, OB_Startup = 197, DB_Any = 208, DB_WWW = 209, DB_Dyn = 210, DB = 257 }

7.8.7.11 ETagListDetails

Beschreibung

Diese Aufzählung enthält alle PLC-Areas, die beim Aktualisieren der Variablentabelle als Filter genutzt werden können.

Tabelle 7- 473 ETagListDetails - Native C++

Syntax	`enum ETagListDetails`
	`{`
	` SRTLD_NONE = 0,`
	` SRTLD_IO = 1,`
	` SRTLD_M = 2,`
	` SRTLD_IOM = 3,`
	` SRTLD_CT = 4,`
	` SRTLD_IOCT = 5,`
	` SRTLD_MCT = 6,`
	` SRTLD_IOMCT = 7,`
	` SRTLD_DB = 8,`
	` SRTLD_IODB = 9,`
	` SRTLD_MDB = 10,`
	` SRTLD_IOMDB = 11,`
	` SRTLD_CTDB = 12,`
	` SRTLD_IOCTDB = 13,`
	` SRTLD_MCTDB = 14,`
	` SRTLD_IOMCTDB = 15`
	`};`

Tabelle 7- 474 ETagListDetails - .NET (C#)

Syntax	`enum ETagListDetails`
	`{`
	` None = 0,`
	` IO = 1,`
	` M = 2,`
	` IOM = 3,`
	` CT = 4,`
	` IOCT = 5,`
	` MCT = 6,`
	` IOMCT = 7,`
	` DB = 8,`
	` IODB = 9,`
	` MDB = 10,`
	` IOMDB = 11,`
	` CTDB = 12,`
	` IOCTDB = 13,`
	` MCTDB = 14,`
	` IOMCTDB = 15`
	`}`

7.8.7.12 ERuntimeConfigChanged

Beschreibung

Diese Aufzählung enthält alle möglichen Ursachen für ein OnConfigurationChanged-Ereignis, das der Runtime Manager sendet.

Tabelle 7- 475 ERuntimeConfigChanged - Native C++

Syntax	`enum ERuntimeConfigChanged`
	`{`
	` SRCC_INSTANCE_REGISTERED = 0,`
	` SRCC_INSTANCE_UNREGISTERED = 1`
	` SRCC_CONNECTION_OPENED = 2,`
	` SRCC_CONNECTION_CLOSED = 3,`
	` SRCC_PORT_OPENED = 4,`
	` SRCC_PORT_CLOSED = 5`
	`};`

Tabelle 7- 476 ERuntimeConfigChanged - .NET (C#)

Syntax	`enum ERuntimeConfigChanged`
	`{`
	` InstanceRegistered = 0,`
	` InstanceUnregistered = 1,`
	` ConnectionOpened = 2,`
	` ConnectionClosed = 3,`
	` PortOpened = 4,`
	` PortClosed = 5`
	`}`

7.8.7.13 EInstanceConfigChanged

Beschreibung

Diese Aufzählung enthält alle möglichen Ursachen für ein OnConfigurationChanged-Ereignis, das der virtuelle Controller sendet.

Tabelle 7- 477 EInstanceConfigChanged - Native C++

Syntax	`enum EInstanceConfigChanged`
	`{`
	` SRICC_HARDWARE_SOFTWARE_CHANGED = 0,`
	` SRICC_IP_CHANGED = 1`
	`};`

Tabelle 7- 478 EInstanceConfigChanged - .NET (C#)

Syntax	`enum EInstanceConfigChanged`
	`{`
	` HardwareSoftwareChanged = 0,`
	` IPChanged = 1`
	`}`

7.8.7.14 EPullOrPlugEventType

Beschreibung

Diese Aufzählung enthält vordefinierte Typen von Ziehen/Stecken-Ereignissen für S7-Module.

Tabelle 7- 479 EPullOrPlugEventType - Native C++

Syntax	`enum EPullOrPlugEventType`
	`{`
	` SR_PPE_UNDEFINED = 0,`
	` SR_PPE_PULL_EVENT = 1,`
	` SR_PPE_PLUG_EVENT = 2,`
	` SR_PPE_PLUG_EVENT_ERROR_REMAINS = 3,`
	` SR_PPE_PLUG_WRONG_MODULE_EVENT = 4`
	`};`

Tabelle 7- 480 EPullOrPlugEventType - .NET (C#)

Syntax	`enum EPullOrPlugEventType`
	`{`
	` Undefined = 0,`
	` Pull = 1,`
	` Plug = 2,`
	` PlugErrorRemains = 3,`
	` PlugWrongModule = 4`
	`}`

7.8.7.15 EProcessEventType

Beschreibung

Diese Aufzählung enthält vordefinierte Typen von Prozessereignissen für S7-Module.

Tabelle 7- 481 EProcessEventType - Native C++

Syntax	`enum EProcessEventType`
	`{`
	` SR_PET_UNDEFINED = 0,`
	` SR_PET_RISING_EDGE = 1,`
	` SR_PET_FALLING_EDGE = 2,`
	` SR_PET_LIMIT1_UNDERRUN = 3,`
	` SR_PET_LIMIT1_OVERRUN = 4,`
	` SR_PET_LIMIT2_UNDERRUN = 5,`
	` SR_PET_LIMIT2_OVERRUN = 6`
	`};`

Tabelle 7- 482 EProcessEventType - .NET (C#)

Syntax	`enum EProcessEventType`
	`{`
	` Undefined = 0,`
	` RisingEdge = 1,`
	` FallingEdge = 2,`
	` Limit_1_Underrun = 3,`
	` Limit_1_Overrun = 4,`
	` Limit_2_Underrun = 5,`
	` Limit_2_Overrun = 6`
	`}`

7.8.7.16 EDirection

Beschreibung

Diese Aufzählung enthält Eigenschaften der Diagnosemeldung.

Tabelle 7- 483 EDirection - Native C++

Syntax	`enum EDirection`
	`{`
	` SRD_DIRECTION_INPUT = 0,`
	` SRD_DIRECTION_OUTPUT = 1`
	`};`

Tabelle 7- 484 EDirection - .NET (C#)

Syntax	`enum EDirection`
	`{`
	` Input = 0,`
	` Output = 1`
	`}`

7.8.7.17 EDiagProperty

Beschreibung

Diese Aufzählung enthält die Kommend-/Gehend-Information der Diagnosemeldung.

Tabelle 7- 485 EDiagProperty - Native C++

Syntax	`enum EDiagProperty`
	`{`
	` SRP_DIAG_APPEAR = 1,`
	` SRP_DIAG_DISAPPEAR = 2`
	`};`

Tabelle 7- 486 EDiagProperty - .NET (C#)

Syntax	`enum EDiagProperty`
	`{`
	` Appear = 1,`
	` Disappear = 2`
	`}`

7.8.7.18 EDiagSeverity

Beschreibung

Diese Aufzählung enthält den Schweregrad der Diagnosemeldung (Fehler, Wartungsanforderung, Wartungsbedarf).

Tabelle 7- 487 EDiagSeverity - Native C++

Syntax	`enum EDiagSeverity`
	`{`
	` SRDS_SEVERITY_FAILURE = 0,`
	` SRDS_SEVERITY_MAINTENANCE_DEMANDED = 1,`
	` SRDS_SEVERITY_MAINTENANCE_REQUIRED = 2`
	`};`

Tabelle 7- 488 EDiagSeverity - .NET (C#)

Syntax	`enum EDiagSeverity`
	`{`
	` Failure = 0,`
	` MaintDemanded = 1,`
	` MaintRequired = 2`
	`}`

7.8.7.19 ERackOrStationFaultType

Beschreibung

Diese Aufzählung enthält die Typen des RackOrStationFault-Ereignisses.

Tabelle 7- 489 ERackOrStationFaultType - Native C++

Syntax	`enum ERackOrStationFaultType`
	`{`
	` SR_RSF_FAULT = 0,`
	` SR_RSF_RETURN = 1`
	`};`

Tabelle 7- 490 ERackOrStationFaultType - .NET (C#)

Syntax	`enum ERackOrStationFaultType`
	`{`
	` Fault = 0,`
	` Return = 1`
	`}`

7.8.7.20 ECycleTimeMonitoringMode

Beschreibung

Diese Aufzählung enthält die Quellen für den Timer zur maximalen Zykluszeitüberwachung.

Tabelle 7- 491 ECycleTimeMonitoringMode - Native C++

Syntax	`enum ECycleTimeMonitoringMode`
	`{`
	` SRCTMM_DOWNLOADED = 0,`
	` SRCTMM_IGNORED = 1,`
	` SRCTMM_SPECIFIED = 2`
	`};`

Tabelle 7- 492 ECycleTimeMonitoringMode - .NET (C#)

Syntax	`enum ECycleTimeMonitoringMode`
	`{`
	` Downloaded = 0,`
	` Ignored = 1,`
	` Specified = 2`
	`}`

7.8.7.21 EAutodiscoverType

Beschreibung

Diese Aufzählung wird bei der Autodiscover Callback-Funktion verwendet.

Tabelle 7- 493 EAutodiscoverType - Native C++

Syntax	`enum EAutodiscoverType`
	`{`
	` SRRSI_DISCOVER_STARTED = 0,`
	` SRRSI_DISCOVER_DATA = 1,`
	` SRRSI_DISCOVER_FINISHED = 2`
	`};`

Tabelle 7- 494 EAutodiscoverType - .NET (C#)

Syntax	`public enum EAutodiscoverType`
	`{`
	` AutodiscoverStarted = 0,`
	` AutodiscoverData = 1,`
	` AutodiscoverFinished = 2`
	`}`

Einschränkungen, Meldungen und Abhilfe 8

8.1 Übersicht

Bestimmte Aktionen oder Ereignisse können in PLCSIM Advanced oder in STEP 7 zu einem Verhalten führen, das von dem einer Hardware-CPU abweicht. Meldungen und mögliche Abhilfen finden Sie in folgenden Kapiteln:

8.2 Einschränkungen bei Fehlersicheren CPUs

Fehler beim Download von Programmänderungen

Beim Download von Programmänderungen auf eine F-CPU kommt es zur Fehlermeldung "korrupt". Die F-CPU bleibt im Betriebszustand STOP, wenn das Optionskästchen "Alle starten" aktiviert ist.

Abhilfe

Deaktivieren Sie im Dialog "Ergebnisse des Ladevorgangs" das Optionskästchen "Alle starten".

Versetzen Sie die CPU nach abgeschlossenem Download manuell über die Schaltfläche RUN in den Betriebszustand RUN.

8.3 OPC UA Server

Mit OPC UA wird ein Datenaustausch über ein offenes, standardisiertes und herstellerunabhängiges Kommunikationsprotokoll durchgeführt. Die CPU als OPC UA Server kann mit OPC UA Clients kommunizieren, z. B. mit HMI-Panels V14 und SCADA-Systemen.

Aus technischen Gründen weichen die Sicherheitseinstellungen in der PLCSIM Advanced von denen einer Hardware-CPU ab. Einige Features sind für Simulationen deaktiviert oder nur eingeschränkt verfügbar.

OPC UA Server konfigurieren

Starten Sie die Instanzen über die Kommunikationsschnittstelle "PLCSIM Virtual Ethernet Adapter" (TCP/IP), um den OPC UA Server zu nutzen.

Die OPC UA Server-Funktionalität ist nicht verfügbar, wenn die Kommunikation über den Softbus erfolgt.

OPC UA Sicherheitseinstellungen

Bezogen auf die OPC UA Sicherheitseinstellungen können in STEP 7 für die Hardware-CPU dieselben Einstellungen vorgenommen werden. PLCSIM Advanced berücksichtigt diese Sicherheitseinstellungen allerdings nicht. So ist gewährleistet, dass der Anwender sein Projekt nicht ändern muss, um eine Simulation durchzuführen.

ACHTUNG
OPC UA Clients OPC UA Clients, die mit PLCSIM Advanced simuliert werden, unterstützen keine Sicherheitseinstellungen (Zertifikate). Aktivieren Sie daher für den OPC UA Server im Register "Eigenschaften" das Kontrollkästchen "Keine Security".

Zertifikate

Hinweis

Server-Zertifikat nicht für sichere Verbindungen nutzen

PLCSIM Advanced nutzt nicht das Zertifikat aus STEP 7, sondern ein eigenes Zertifikat in der Firmware. Das Zertifikat muss für Simulationen nicht geändert werden. Es besitzt aber nicht dieselbe Sicherheitsstufe wie ein heruntergeladenes Server-Zertifikat und kann nicht für sichere Verbindungen benutzt werden!

- **Server Security Endpoints**

  PLCSIM Advanced unterstützt nur den Security Endpoint "none".

- **Client-Zertifikate**

  PLCSIM Advanced verwertet nicht die importierten und in STEP 7 eingestellten Zertifikate. PLCSIM Advanced akzeptiert alle Client-Zertifikate automatisch. Diese Voreinstellung kann nicht geändert werden.

- **Benutzer-Authentifizierung**

  PLCSIM Advanced übernimmt nicht die in STEP 7 eingestellten Benutzernamen.

  Es ist nur ein Login als "guest" oder "anonymous" möglich.

Nutzungsberechtigung für OPC UA

Die PLCSIM Advanced Lizenz enthält auch die Nutzungsberechtigung für OPC UA.

Die Nutzungsberechtigung gilt für zwei Instanzen.

8.4 Webserver

Der in eine CPU integrierte Webserver ermöglicht die Überwachung und Verwaltung der CPU durch berechtigte Nutzer über ein Netzwerk. Auswertungen und Diagnose sind somit über große Entfernungen möglich.

Jede PLCSIM Advanced Instanz kann ihren eigenen Webserver simulieren.

Die Simulation des Webservers ist unter S7-PLCSIM Advanced V3.0 eingeschränkt:

- Das Sichern und Wiederherstellen einer Projektierung über den Webserver ist nicht möglich.
- Der Freeze-Zustand eines virtuellen Controllers wird als interner Betriebszustand nicht angezeigt.

Webserver konfigurieren

S7-PLCSIM Advanced

Starten Sie die Instanzen über die Kommunikationsschnittstelle "PLCSIM Virtual Ethernet Adapter" (TCP/IP), um den Webserver zu nutzen.

Die Webserver-Funktionalität ist nicht verfügbar, wenn die Kommunikation über den Softbus erfolgt.

STEP 7

Konfigurieren Sie den Webserver in STEP 7 in den Eigenschaften der CPU.

Eingeschränkte Webserver-Funktionalität

- Login

  Als Benutzer ist "PLCSIM" voreingestellt. Es gibt kein Login für Anwender.

  Ein in STEP 7 eingestellter Benutzer und seine Rechte haben keinen Einfluss auf den Benutzer "PLCSIM".
- Es gibt keinen Zugriff über das sichere Übertragungsprotokoll "HTTPS".
- Auf einigen Webseiten werden aufgrund von unterschiedlichem Datenhandling die Informationen nicht vollständig angezeigt.
- Es gibt keine Topologie-Informationen.
- "Online Backup&Restore" ist nicht verfügbar.
- FW-Updates werden nicht unterstützt.

Anzahl der maximalen Verbindungen

Im Webserver wird ein fester Wert von 384 für die maximal möglichen Verbindungen angezeigt.

Abhilfe

Entnehmen Sie den richtigen Wert aus den technischen Daten zur geladenen CPU.

8.5 Projektierung einer PLCSIM Advanced Instanz sichern und wiederherstellen

Projektierung sichern und wiederherstellen

Ab PLCSIM Advanced V2.0 ist es möglich, eine PLCSIM Advanced **Instanz** zu sichern und wiederherzustellen.

Sie können beliebig viele Sicherungen anlegen und so unterschiedliche Projektierungen für eine PLCSIM Advanced Instanz vorhalten.

Das Sichern und Wiederherstellen führen Sie im TIA Portal so durch wie bei einer realen CPU.

Das Sichern und Wiederherstellen über Webserver und (simuliertes) Display wird nicht unterstützt.

Eine Sicherung, die mit PLCSIM Advanced erstellt wurde, kann nur mit PLCSIM Advanced benutzt werden.

Das Wiederherstellen der Projektierung einer realen CPU ist nicht mit einer Sicherung aus PLCSIM Advanced möglich.

Voraussetzung

- Das Sichern und Wiederherstellen der Projektierung einer PLCSIM Advanced Instanz erfolgt über das Protokoll TCP/IP, Softbus wird nicht unterstützt.
- Das Wiederherstellen der Projektierung einer PLCSIM Advanced Instanz ist nur mit der entsprechenden Sicherung aus PLCSIM Advanced möglich.

8.6 Einschränkungen bei Dateipfaden

Für Anwenderschnittstellen, die als Übergabeparameter einen Pfad oder einen vollständigen Dateinamen erwarten, gelten folgende Einschränkungen:

Einschränkungen bei lokalen Pfaden	Schreibrechte auf systemkritische Verzeichnisse, wie das Windows-Verzeichnis (%Windows%) oder die Programmverzeichnisse (%Program Files%, %Program Files (x86)%) sind nicht erlaubt. Die C++ Anwenderschnittstelle liefert in diesem Fall den Fehlercode `SREC_WRONG_ARGUMENT` zurück. Die managed Anwenderschnittstelle gibt in diesem Fall eine Ausnahme mit dem Fehlercode `RuntimeErrorCode.WrongArgument` zurück.
Einschränkungen bei Netzwerkpfaden	Um Netzwerkpfade nutzen zu können, müssen Sie diese als Netzlaufwerk einbinden. Ansonsten liefert die C++ Anwenderschnittstelle den Fehlercode `SREC_WRONG_ARGUMENT` zurück. Die managed Anwenderschnittstelle gibt eine Ausnahme mit dem Fehlercode `RuntimeErrorCode.WrongArgument` zurück.

8.7 Einschränkungen bei Kommunikationsdiensten

TUSEND / TURCV

Wenn Sie die UDP-Bausteine TUSEND und TURCV über die Kommunikationsschnittstelle "PLCSIM" (Softbus) ausführen, dann erhalten Sie auf Sendeseite und Empfangsseite den Fehlercode 0x80C4:

Temporary communications error. The specified connection is temporarily down.

Abhilfe

Stellen Sie in PLCSIM Advanced die Kommunikationsschnittstelle "PLCSIM Virtual Ethernet Adapter" (TCP/IP) ein.

Sichere TCP-Verbindungen

PLCSIM Advanced unterstützt TLS (Transport Layer Security) nicht und hat kein Zertifikats-Management. Deshalb können aus der Simulation heraus keine sicheren TCP-Verbindungen aufgebaut werden.

Das bedeutet, dass folgende Verbindungen nicht unterstützt werden:

- Sichere OUC-Verbindungen (Secure Open User Communication)
- Gesicherte Verbindungen zu einem Mailserver mit TMAIL_C
- HTTPS-Verbindungen zum Webserver

8.8 Einschränkungen bei Anweisungen

PLCSIM Advanced simuliert Anweisungen für die CPUs S7-1500 und ET 200SP so realitätsnah wie möglich. PLCSIM Advanced prüft die Eingangsparameter auf Gültigkeit und gibt Ausgänge zurück, die zwar gültig sind, jedoch nicht unbedingt denen entsprechen, die eine reale CPU mit physischen Eingängen / Ausgängen zurückgeben würde.

Nicht unterstützte Anweisungen

Nicht unterstützte Anweisungen werden als nicht betriebsbereit behandelt, ihr Wert ist immer "OK". PLCSIM Advanced unterstützt die folgenden Anweisungen nicht:

- DP_TOPOL
- PORT_CFG

8.9 Einschränkungen bei lokaler Kommunikation über Softbus

Identische IP-Adressen für Instanzen

Wenn die Kommunikationsschnittstelle "PLCSIM" (Softbus) eingestellt ist, dann werden beim Erstellen der Instanzen über das Control Panel automatisch für alle Instanzen identische IP-Adressen erstellt.

In STEP 7 wird in der Lifelist daher nur eine Instanz angezeigt.

Abhilfe

Weisen Sie über die API-Funktion `SetIPSuite()` jeder Instanz eine eindeutige Adresse zu, dann werden in STEP 7 alle Instanzen mit ihren IP-Adressen angezeigt.

API-Funktion

- SetIPSuite() (Seite 161)

Arbeiten mit multiplen Instanzen

Wenn Sie mit Instanzen **ohne eindeutige IP-Adressen** arbeiten, dann beachten Sie folgende Vorgehensweise beim Download aus dem TIA Portal über "PLCSIM" (Softbus):

1. Starten Sie im Control Panel **nur eine** Instanz mit dem Symbol 
2. Führen Sie aus dem TIA Portal den Programm Download auf diese Instanz durch.
3. Wiederholen Sie die Schritte, bis Sie alle Instanzen erzeugt und alle Projekte heruntergeladen haben.

Online und Diagnose

Wenn die Kommunikationsschnittstelle "PLCSIM" (Softbus) eingestellt ist, dann werden bei der Funktion "Online und Diagnose" unter PROFINET-Schnittstelle keine Details angezeigt (IP-Adresse, MAC-Adresse...).

Siehe auch

Controller - Informationen und Einstellungen (Seite 157)

8.10 Meldungen bei Kommunikation über TCP/IP

Fehlercodes

Wenn in der Taskleiste eine ID mit Fehlerbezeichnung erscheint, dann finden Sie die Beschreibung dazu im Kapitel 7 "Anwenderschnittstellen (API)".

Bild 8-1 Beispiel: Fehlercode 63

Meldungen und Abhilfe

Im S7-PLCSIM Advanced Control Panel werden die Einstellungen zur TCP/IP-Kommunikation geprüft. Im Folgenden finden Sie die Meldungen und die entsprechenden Abhilfen:

Meldung

"Siemens PLCSIM Virtual Ethernet Adapter was not found. Please reinstall PLCSIM Advanced."

Abhilfe

Der PLCSIM Virtual Ethernet Adapter kann auf dem System nicht gefunden werden.

Führen Sie das PLCSIM Advanced Setup erneut aus:

1. Doppelklicken Sie auf das Download-Paket oder legen Sie das Installationsmedium in das Laufwerk ein. Das Setup-Programm startet automatisch, sofern Sie nicht die Autostartfunktion auf dem Computer deaktiviert haben. Wenn das Setup-Programm nicht automatisch startet, starten Sie es manuell durch Doppelklick auf die Datei "Start.exe".
2. Befolgen Sie die Eingabeaufforderungen, bis Sie in das Fenster "Konfiguration" gelangen. Wählen Sie das Optionskästchen "Reparieren".
3. Befolgen Sie die weiteren Eingabeaufforderungen, um Ihre Installation zu reparieren.
4. Um den Reparaturvorgang abzuschließen, starten Sie Ihren Computer neu.

Meldung

"Siemens PLCSIM Virtual Ethernet Adapter is disabled. Please enable it."

Abhilfe

Der PLCSIM Virtual Ethernet Adapter ist auf dem System deaktiviert. Öffnen Sie in der Systemsteuerung unter "Netzwerk- und Freigabecenter" > "Adaptereinstellungen ändern" und aktivieren Sie den Netzwerkadapter.

Meldung

"NetGroup Packet Filter Driver (NPF) is not running. Start it from cmd with 'net start npf'."

Abhilfe

Der NetGroup Packet Filter Driver (NPF) ist auf dem System nicht aktiv. Starten Sie die Kommandozeile im Administrator-Modus und führen Sie den Befehl "net start npf" aus.

Meldung

"You have to set a valid IP address for the Siemens PLCSIM Virtual Ethernet Adapter."

Abhilfe

Weisen Sie dem Siemens PLCSIM Virtual Ethernet Adapter eine statische IP-Adresse zu oder beziehen Sie eine IP-Adresse über DHCP (Voreinstellung).

8.11 Einschränkung der Sicherheit bei VMware vSphere Hypervisor (ESXi)

Wenn Sie die Virtualisierungsplattform VMware vSphere Hypervisor (ESXi) verwenden, müssen Sie die Richtlinienausnahmen ändern, um die Kommunikation über TCP/IP nutzen zu können.

Abhilfe

Akzeptieren Sie für den Virtual Switch des ESXi die Optionen "Promiscuous-Modus" und "Gefälschte Übertragungen".

ACHTUNG
Einschränkung der Sicherheit Aus Sicherheitsgründen ist der Promiscuous-Modus standardmäßig ausgeschaltet. Wenn Sie den Promiscuous-Modus akzeptieren, dann empfängt der reale Ethernet Adapter auch Telegramme, die nicht an ihn adressiert sind.

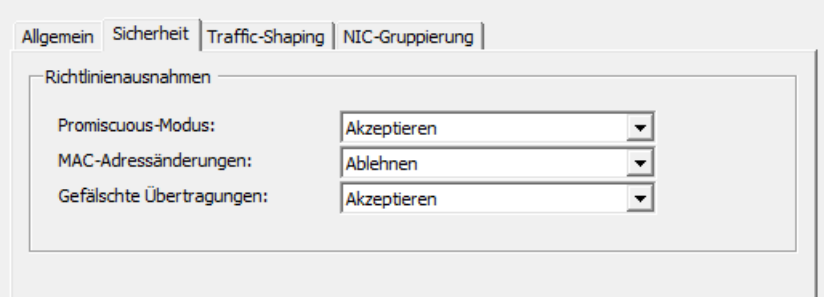


Bild 8-2 Richtlinienausnahmen für VMware vSphere Hypervisor (ESXi)

8.12 Überwachung Überlauf

Überwachung Hauptzyklus

Die maximale Zykluszeitüberwachung beträgt für PLCSIM Advanced eine Minute.

Wenn Sie die Werte verwenden möchten, die im TIA Portal projektiert sind, dann stellen Sie diese über die folgende API-Funktion ein: `SetCycleTimeMonitoringMode()`.

Siehe Zykluskontrolle (Seite 248)

Überwachung zyklischer Ereignisse

Wenn Ihre Simulation Weckalarme enthält, kann die Warteschlange von PLCSIM Advanced für zyklische Ereignisse überlaufen. Aufgrund der Ablaufgeschwindigkeit von PLCSIM Advanced im Vergleich zu realer Hardware kann die benötigte Zeit zur Erstellung des Diagnosepuffereintrags länger sein als die Zeit bis zum nächsten Weckalarm.

Wenn dies der Fall ist, wird ein zusätzlicher Eintrag in die Warteschlange gestellt, der einen weiteren Überlauf verursacht. PLCSIM Advanced gibt bei einem Überlauf visuelle Hinweise in Form von Diagnosepuffermeldungen und einem roten Fehlersymbol im Projektbaum aus.

Siehe auch

Simulation beschleunigen und verlangsamen (Seite 90)

8.13 Abweichende E/A-Werte im STEP 7-Anwenderprogramm

Aktualisierte Werte

Jede Wertänderung, die ein STEP 7-Anwenderprogramm in den E/A-Adressbereichen vornimmt, wird im Zykluskontrollpunkt mit dem aktualisierten Wert überschrieben, der über die API-Funktionen `Write...()` geschrieben wurde. Die API-Funktionen `Read...()` liefern **für den Eingangsbereich** nur diesen aktualisierten Wert und nicht den Wert aus STEP 7 zurück.

Nicht aktualisierte Werte

Wenn der Wert über die API-Funktionen `Write...()` nicht aktualisiert wurde, liefern die API-Funktionen `Read...()` **für den Ausgangsbereich** den Wert aus STEP 7 zurück.

Siehe auch

Peripherie-I/O simulieren (Seite 74)

8.14 Mehrfache Simulationen und mögliche Kollision der IP-Adressen

Sie können gleichzeitige mehrere CPUs simulieren, aber jede simulierte CPU-Schnittstelle benötigt eine eindeutige IP-Adresse.

Stellen Sie sicher, dass Ihre CPUs unterschiedliche IP-Adressen haben, bevor Sie die Simulationen starten.

8.15 Fehlender Zugriff auf eine IP-Adresse

Besonderheit bei der verteilten Kommunikation

Wenn Sie mehrere Netzwerk-Teilnehmer im selben Subnetz über unterschiedliche virtuelle oder reale Adapter nutzen, kann es vorkommen, dass das Betriebssystem den Teilnehmer am falschen Adapter sucht.

Abhilfe

Wiederholen Sie Ihre Zugriffe oder geben Sie im Kommandozeilen-Editor von Windows "arp -d <IP-Adresse>" ein.

8.16 Simulation im Standby-Modus

Wenn Ihr Computer oder Programmiergerät in den Standby- oder Ruhemodus wechselt, wird die Simulation möglicherweise angehalten. In diesem Fall wird auch die Kommunikation zwischen STEP 7 und PLCSIM Advanced angehalten. Wird Ihr Computer oder Programmiergerät wieder aktiviert, muss die Kommunikation gegebenenfalls erneut hergestellt werden. In einigen Fällen kann es auch erforderlich sein, das Simulationsprojekt erneut zu öffnen.

Um diese Situation zu verhindern, deaktivieren Sie den Standby-Modus Ihres Computers oder Programmiergeräts.

Liste der Abkürzungen

A

Abkürzung	Begriff
ALM	Automation License Manager Werkzeug zur Verwaltung von License Keys in STEP 7
API	Application Programming Interface - Anwenderschnittstelle
arp	Address resolution protocol
BCD	Binary Coded Decimal
CPU	Central Processing Unit (Synonym für PLC)
DLL	Dynamic Link Library
HMI	Human Machine Interface - Benutzerschnittstelle
IE	Industrial Ethernet
GUI	Graphical User Interface
LAN	Local Area Network Computernetzwerk, das auf einen begrenzten örtlichen Bereich beschränkt ist.
MFP	Multifunktionale Plattform
OB	Organization Block
ODK	Open Development Kit
OPC UA	Open Platform Communications Unified Architecture
PG	Programmiergerät
PLC	Programmable Logic Controller
PN	PROFINET
RAM	Random Access Memory
RT	Runtime
SO	Shared Object
TCP/IP	Transmission Control Protocol/Internet Protocol
TIA	Totally Integrated Automation
TPA	Teilprozessabbild
UTC	Coordinated Universal Time
VM	Virtual Machine
VPLC	Virtual Programmable Logic Controller
WinCC	Windows Control Center