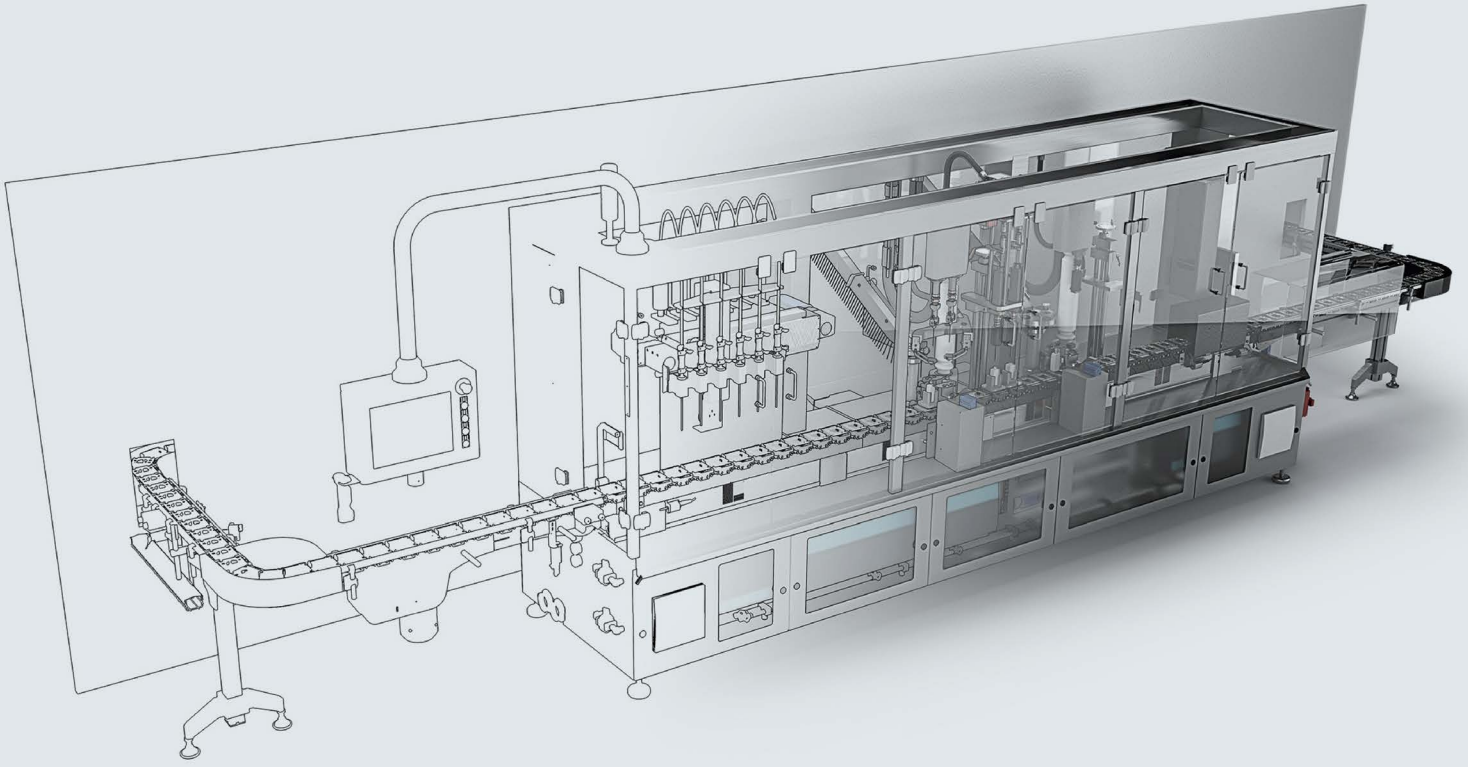
SIEMENS



SIMATIC

S7-1500

S7-PLCSIM Advanced

Funktionshandbuch

Ausgabe 09/2016

siemens.com

SIEMENS

SIMATIC

S7-1500
S7-PLCSIM Advanced

Funktionshandbuch

09/2016
A5E37039506-AA

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
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
DEUTSCHLAND

A5E37039506-AA
℗ 08/2016 Änderungen vorbehalten

Vorwort

Zweck der Dokumentation

Das vorliegende Funktionshandbuch beschreibt die Simulations-Software S7-PLCSIM Advanced V1.0. Sie können damit Ihre CPU-Programme auf einem simulierten, virtuellen S7-1500 Controller testen.

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

Gültigkeitsbereich

Das Funktionshandbuch ist gültig für folgende Bestellvarianten:

- 6ES7823-1FA00-0YA5 - S7-PLCSIM Advanced V1.0 DVD
- 6ES7823-1FE00-0YA5 - S7-PLCSIM Advanced V1.0 Download

Die Artikel beinhalten jeweils eine Lizenz für eine Instanz.

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
- Entwicklung von Software in C++ und 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 V1.0 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.

Security-Hinweise

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

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

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

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

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

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

Inhaltsverzeichnis

Tabellen

Bilder

Wegweiser 1

1.1 Wegweiser Dokumentation S7-PLCSIM Advanced

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 (http://w3.siemens.com/mcms/industrial-automation-systems-simatic/de/handbuchuebersicht/Seiten/Default.aspx).

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).

1.2 S7-PLCSIM Produkte

S7-PLCSIM Advanced

Die Version von S7-PLCSIM Advanced unterstützt folgende SIMATIC CPU-Familien:

- S7-1500, S7-1500F, S7-1500C, S7-1500T und S7-1500TF
- ET 200SP und ET 200SP F

Kommunikation

S7-PLCSIM Advanced unterstützt die Kommunikation über Softbus oder TCP/IP.

S7-PLCSIM Advanced ermöglicht C++ und C#-Programmen und Simulations-Software den Zugriff über die Anwenderschnittstelle (API) auf die von S7-PLCSIM Advanced unterstützten SIMATIC CPU-Familien.

S7-PLCSIM Advanced und S7-PLCSIM ab V12 können nicht auf dem selben PC oder der selben virtuellen Maschine installiert werden. Die Kommunikation zwischen beiden Anwendungen ist nicht simulierbar.

S7-PLCSIM V5.4 SP7 wird automatisch mit S7-PLCSIM Advanced installiert. Die Kommunikation zwischen beiden Anwendungen ist simulierbar.

S7-PLCSIM V14

Die Version von S7-PLCSIM unterstützt folgende SIMATIC CPU-Familien:

- S7-1200 und S7-1200F
- S7-1500, S7-1500F, S7-1500C, S7-1500T und S7-1500TF
- ET 200SP und ET 200SP F

Kommunikation

S7-PLCSIM V14 unterstützt nur die Kommunikation über Softbus.

S7-PLCSIM V5.x

S7-PLCSIM V5.x simuliert folgende SIMATIC CPU-Familien:

- S7-300 und S7-300F
- S7-400 und S7-400F

S7-PLCSIM V5.x kann auf dem selben PC oder der selben virtuellen Maschine installiert und betrieben werden wie S7-PLCSIM ab V12.

S7-PLCSIM V5.4 SP7 wird automatisch mit S7-PLCSIM Advanced installiert.

Kommunikation

S7-PLCSIM V5.x kann über Softbus mit Instanzen von S7-PLCSIM ab V12 kommunizieren.

S7-PLCSIM V5.4 SP7 kann über Softbus mit Instanzen von S7-PLCSIM Advanced kommunizieren.

Produktübersicht 2

2.1 Was ist PLCSIM Advanced?

Simulationssysteme unterstützen die Entwicklung von Programmen und den daran anschließenden produktiven Einsatz. In der Automatisierungswelt verkürzt eine simulierte Testumgebung die Inbetriebnahmezeiten. Bei Programmänderungen ist es möglich, das Programm im virtuellen Controller zu testen, bevor es in die entsprechende reale Steuerung geladen und die Anlage in Betrieb gesetzt wird.

S7-PLCSIM Advanced

Mit S7-PLCSIM Advanced können Sie Ihre CPU-Programme auf einem virtuellen Controller simulieren. Sie benötigen dafür keine echten Controller. Sie können Ihre CPU in STEP 7 V14 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 S7-PLCSIM Advanced einen vollwertigen Ethernet-Anschluss und kann somit auch verteilt kommunizieren.

Anwenderschnittstelle (API)

S7-PLCSIM Advanced ermöglicht über die Anwenderschnittstelle die Interaktion mit eigenen C++/C#-Programmen oder Software.

Anwendungsgebiete

- Software in the Loop-Simulation zur virtuellen Inbetriebnahme von Werkzeugmaschinen
- In Verbindung mit Software von Drittanbietern:
  - Simulation von Produktionsmaschinen und Anlagen
  - Kombinierte Simulation von Automation und Mechanik

Vorteile

Der Einsatz von S7-PLCSIM Advanced bietet zahlreiche Vorteile:

- Qualität der Automatisierungsprojekte verbessern
- Markteinführungszeiten verkürzen
- Fertigungszeiten reduzieren
- Risiken bei der Inbetriebnahme reduzieren
- Kosten für Hardware in Simulationsumgebungen vermeiden
- Effizienz bei der Wartung steigern

2.2 Sicherheit bei 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 Komponente WinPCap bietet 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

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

2.3 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.

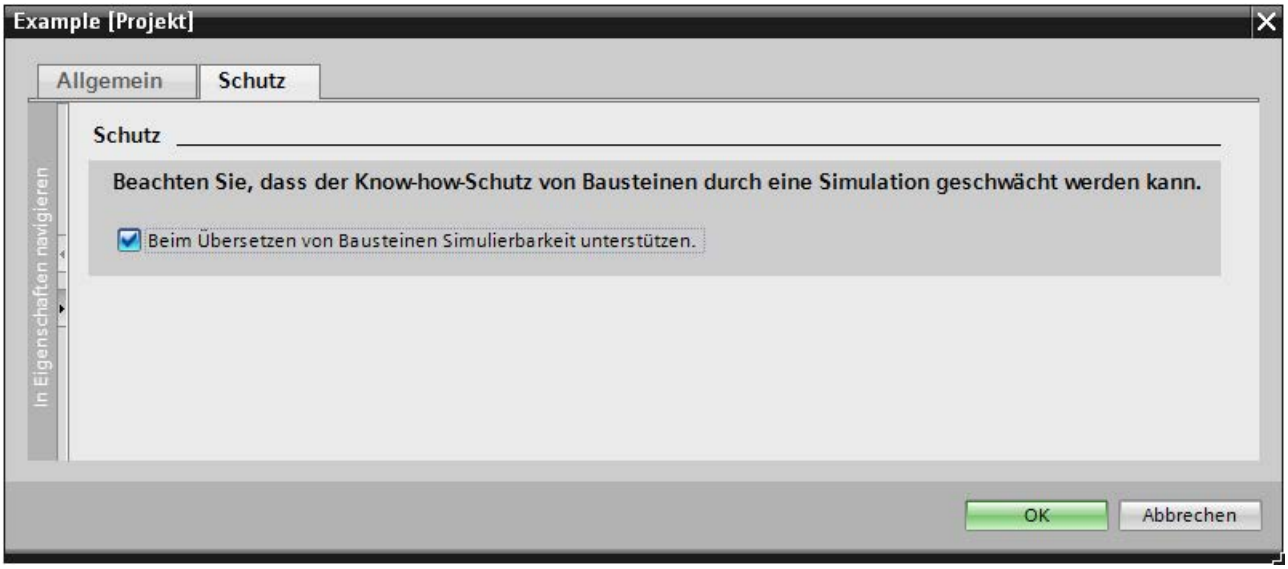


Bild 2-1 Simulierbarkeit aktivieren

Know-how-Schutz

Wenn ein know-how-geschützter Baustein für die Simulation verwendet werden soll, muss er durch Passworteingabe entsperrt werden, damit in den Eigenschaften des Bausteins im Register "Allgemein > Übersetzung" die Option "Simulierbarkeit mit SIMATIC S7-PLCSIM Advanced" aktiviert werden kann.

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).

2.4 Unterstützte CPUs

Unterstützte CPUs aus der S7-1500-Familie

S7-PLCSIM Advanced V1.0 unterstützt die Simulation folgender CPUs (alle Firmware-Versionen, empfohlen wird V2.0):

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

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

2 Die ODK-Funktionalität dieser CPU wird nicht simuliert.

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

4 Die Simulation dieser CPU unterstützt nur 64 Kurvenscheiben.

Nicht unterstützte CPUs

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

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

2.5 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

S7-PLCSIM Advanced wird auf einem PC mit dem Betriebssystem Windows ausgeführt. Daher sind die Scanzykluszeit und die genaue Zeit von Aktionen in S7-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.

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 geschützten Bausteinen können nur simuliert werden, wenn sie für die Simulation freigeschaltet sind. Hierfür benöigen Sie das Passwort des Bausteins.

Anweisungen

Anweisungen werden bis auf wenige Ausnahmen, z. B. Alarme, simuliert. Programme, die sich auf die entsprechenden Anweisungen stützen, verhalten sich bei der Simulation anders als bei realen CPUs.

Siehe auch

Einschränkungen bei Anweisungen (Seite 320)

2.5.1 Einschränkungen bei allen unterstützten CPUs

Peripherie

S7-PLCSIM Advanced simuliert die reale CPU, nicht aber projektierte Peripheriemodule und die Onboard-Peripherie der Kompakt-CPUs.

Bussysteme

S7-PLCSIM Advanced simuliert keine Bussysteme (PROFINET IO, PROFIBUS DP, Rückwandbus).

Teilprozessabbilder

S7-PLCSIM Advanced aktualisiert die Adressbereiche des Prozessabbilds immer am Zykluskontrollpunkt. Teilprozessabbilder werden zwischen zwei Zykluskontrollpunkten nicht aktualisiert.

Diagnose

S7-PLCSIM Advanced kann nicht den kompletten Umfang aller möglichen Diagnosepuffereinträge simulieren, z. B. keine Einträge, die die reale Hardware betreffen. Die meisten E/A- und Programmfehler werden jedoch simuliert.

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. S7-PLCSIM Advanced simuliert diese Funktion nicht.

Datenprotokollierung

S7-PLCSIM Advanced simuliert die Datenprotokollierung nicht, weil diese Funktion alle Ausgänge auf die SD-Speicherkarte schreibt und S7-PLCSIM Advanced die Verwendung einer SD-Speicherkarte nicht unterstützt.

Rezepte

S7-PLCSIM Advanced simuliert die Verwendung von Rezepten nicht.

Backup und Restore

S7-PLCSIM Advanced simuliert die Produkteigenschaft "Backup und Restore" nicht.

Eingeschränkte Unterstützung

S7-PLCSIM Advanced simuliert einige Funktionen in eingeschränktem Umfang. Eine Übersicht finden Sie im Kapitel Einschränkungen (Seite 316).

2.6 Motion Control simulieren

Einschränkungen

PLCSIM Advanced unterstützt STEP 7-Projekte mit Konfigurationen und Funktionen für Motion Control für die CPUs S7-1500, S7-1500F, S7-1500C, S7-1500T, S7-1500T F, ET 200SP und ET 200SP F.

PLCSIM Advanced unterstützt Technologieobjekte bei einigen CPUs nur eingeschränkt, siehe Einschränkungen bei Motion Control (Seite 321).

Simulation mit externer Simulations-Software

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

Simulations-Modus in STEP 7

Der Simulations-Modus ist eine Standardfunktion der Technologieobjekte.

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).

Installieren

3

3.1 Einleitung

3.1.1 Systemanforderungen

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

Mindestvoraussetzungen für Hardware und Software

PLCSIM Advanced installieren Sie bevorzugt auf einem PC stand-alone, unabhängig von STEP 7. Alternativ können Sie PLCSIM Advanced auf dem Projektierungs-PC installieren, auf dem bereits STEP 7 installiert ist.

Damit PLCSIM Advanced effizient funktioniert, müssen Mindestvoraussetzungen bezüglich der Computer-Hardware und -Software gegeben sein.

Tabelle 3- 1 Mindestvoraussetzungen für Hardware und Software

Hardware / Software	Voraussetzung
Prozessor	2,2 GHz Intel® Celeron® Dual Core
RAM	• 4 GB für eine Instanz • 8 GB für 4 Instanzen
Freier Festplattenspeicher	5 GB
Betriebssystem 64 Bit-Variante	• Windows 7 Home Premium SP1 • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 • Windows Server 2012 R2 StdE (vollständige Installation)
Bildschirmauflösung	1024 x 768

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) 6.0
- VMware Workstation 12 Pro
- VMware Workstation Player 12
- Microsoft Windows Server 2012 R2 Hyper-V

Gast-Betriebssysteme

Die folgenden Gast-Betriebssysteme können Sie auf der ausgewählten Virtualisierungsplattform für die Installation von STEP 7 V14 und PLCSIM Advanced verwenden:

- Windows 7 Professional SP1 / Ultimate SP1 / Enterprise SP1 (64-Bit)
- Windows Server 2012 R2 (64-Bit)

3.1.2 Lizenzen

Floating License

S7-PLCSIM Advanced wird mit einer Lizenz vom Typ Floating ausgeliefert. Diese kann lokal abgelegt und für ein Netzwerk freigegeben werden.

Hinweis

Eine Floating License ist gültig für **eine** Instanz eines virtuellen Controllers innerhalb einer PLCSIM Advanced Installation.

Umgang mit Lizenzen

Den Umgang mit den Lizenzen für S7-PLCSIM Advanced Instanzen finden Sie auf der DVD in der Beschreibung SIMATIC Automation License Manager (ALM).

API-Funktionen zu Lizenzen

- Rückgabewerte für die API-Funktion PowerOn() (Seite 134) und die Callback-Funktion OnOperatingStateChanged (Seite 239)
  - `SREC_OK`, wenn eine Lizenz verfügbar ist.
  - `SREC_WARNING_TRIAL_MODE_ACTIVE`, wenn keine Lizenz verfügbar ist und eine Instanz in einem Modus gestartet ist, der für eine Stunde die uneingeschränkte Nutzung der Instanz erlaubt.
- Rückgabewert für die Callback-Funktion `OnOperatingStateChanged`
  - `SREC_LICENSE_NOT_FOUND`, wenn die Instanz nach Ablauf des Trial Modes automatisch abgeschaltet wird.

Warnung trotz vorhandener Lizenz

Hinweis

Ob eine Lizenz vorhanden ist, wird beim Hochlauf und nach einer Stunde überprüft, in der Sie eine Instanz ohne Einschränkung testen können (Trial Mode). Wenn Sie in der Zwischenzeit eine Lizenz einspielen, wird daher dennoch `SREC_WARNING_TRIAL_MODE_ACTIVE` zurückgegeben.

3.1.3 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_V01@<DATUM_UHRZEIT>.log
- SIA_S7-PLCSIM_Advanced_V01@<DATUM_UHRZEIT>_summary.log

Setup_Report (CAB-Datei)

Eine Archivdatei mit dem Installationsprotokoll und allen anderen erforderlichen Dateien wird im CAB-Format gespeichert. Diese Archivdatei finden Sie unter "%autinstlog%\Reports\Setup_report.cab".

Wenn Sie Hilfe bei der Installation benötigen, dann senden Sie dem technischen Support von Siemens diese CAB-Datei. Der technische Support kann die Fehlerbehebung Ihrer Installation anhand der Informationen in der CAB-Datei durchführen.

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

3.2 Installationsmedium

Bewahren Sie das Installationsmedium nach der Installation von S7-PLCSIM Advanced an einem sicheren, leicht zugänglichen Ort auf.

Mit dem Installationsmedium können Sie Ihre Installation bei Bedarf ändern, reparieren oder deinstallieren.

3.3 PLCSIM Advanced installieren

S7-PLCSIM Advanced beginnt automatisch mit der Installation, wenn Sie das Installationsmedium in Ihr DVD-Laufwerk einlegen.

Voraussetzung für die Installation

Vergewissern Sie sich, dass die folgenden Bedingungen erfüllt sind, bevor Sie mit dem Installationsvorgang beginnen:

- Hardware und Software des PCs oder des Siemens Field PGs erfüllen die Systemvoraussetzungen.
- Die Person, die die Installation durchführt, hat auf dem jeweiligen Computer Administratorrechte.
- Keine anderen Programme sind aktiv. Dies gilt auch für den Siemens Automation License Manager und andere Anwendungen von Siemens.
- Alle S7-PLCSIM Versionen ab V12 sind deinstalliert.

Hinweis

Sicherheitseinstellungen

Für die Lizenzierung über den ALM müssen Sie bei der Installation von PLCSIM Advanced zustimmen, dass der Port 4410 für TCP als Ausnahme in der Windows-Firewall eingegeben wird (Schritt 7).

S7-PLCSIM Advanced installieren

Gehen Sie zur Installation von PLCSIM Advanced wie folgt vor:

1. Legen Sie das Installationsmedium in das DVD-Laufwerk Ihres Computers ein. Das Setup-Programm wird automatisch gestartet, sofern Sie nicht die Autostartfunktion auf dem Field PG oder PC deaktiviert haben. Wird das Setup-Programm nicht automatisch gestartet, starten Sie es manuell durch Doppelklick auf die Datei "Start.exe". Das Fenster "General settings" wird angezeigt.
2. Das Optionskästchen für die Sprache Englisch ist aktiv.
3. Klicken Sie auf die Schaltfläche "Installationshinweise lesen", um Installationsinformationen anzuzeigen. Wenn Sie die Hinweise gelesen haben, schließen Sie die Datei.
4. Klicken Sie auf die Schaltfläche "Produktinformationen lesen", um die "Liesmich-Datei" anzuzeigen. Wenn Sie die Informationen gelesen haben, schließen Sie die Datei.
5. 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 Standard-Installationspfad auswählen, wird das Desktop-Symbol möglicherweise nicht richtig angezeigt.
6. 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.
7. 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".
8. Im Übersichtsbildschirm werden Ihre Installationsdetails angezeigt. Klicken Sie auf die Schaltfläche "Installieren". Daraufhin wird die Installation gestartet und PLCSIM Advanced auf Ihrem Computersystem installiert.
9. 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.
10. Klicken Sie auf "Neu starten". Wenn der Computer nicht neu gestartet wird, klicken Sie auf "Beenden".

Fehler beim Installieren von S7-PLCSIM Advanced

Bei der Installation von S7-PLCSIM Advanced wird eine vorhandene Installation von S7-PLCSIM angezeigt.

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

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

Abhilfe

Führen Sie für S7-PLCSIM V12 oder V12 SP1 das Setup durch und deinstallieren Sie 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 PLCSIM Advanced ändern

Voraussetzungen zum Ändern der Installation

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 zum Ändern der Installation

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

1. Legen Sie das Installationsmedium in das Laufwerk ein. Das Setup-Programm startet automatisch, sofern Sie nicht die Autostartfunktion auf dem Field PG oder PC 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 PLCSIM Advanced reparieren

Falls Ihre Installation beschädigt wird, können Sie sie mit dem PLCSIM Advanced-Datenträger reparieren.

Voraussetzungen zum Reparieren der Installation

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 zum Reparieren der Installation

Gehen Sie zum Reparieren Ihrer Installation wie folgt vor:

1. Legen Sie das Installationsmedium in das Laufwerk ein. Das Setup-Programm wird automatisch gestartet, sofern Sie nicht die Autostartfunktion auf dem Field PG oder PC deaktiviert haben. Wird das Setup-Programm nicht automatisch gestartet, 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 PLCSIM Advanced deinstallieren

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

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

PLCSIM Advanced über die Windows-Systemsteuerung deinstallieren

Gehen Sie zum Deinstallieren von PLCSIM Advanced über die Windows-Systemsteuerung 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 V1.0" 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 bei der Deinstallation von PLCSIM Advanced über die Windows-Systemsteuerung Probleme auftreten, verwenden Sie zum Deinstallieren der Anwendung das Installationsmedium.

PLCSIM Advanced über das Installationsmedium deinstallieren

Gehen Sie zum Deinstallieren von PLCSIM Advanced über das Installationsmedium wie folgt vor:

1. Legen Sie das Installationsmedium in das Laufwerk ein. Das Setup-Programm wird automatisch gestartet, sofern Sie nicht die Autostartfunktion auf dem Programmiergerät oder PC deaktiviert haben. Wenn das Setup-Programm nicht automatisch gestartet wird, dann 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 von PLCSIM Advanced weiterhin die Eingabeaufforderungen.
4. Um den Deinstallationsvorgang abzuschließen, starten Sie Ihren Computer neu.

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

Weitere Software deinstallieren

Bei der Deinstallation von PLCSIM Advanced bleibt folgende Software 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 V14 und den Instanzen der PLCSIM Advanced Anwenderschnittstellen stehen folgende Wege offen:

Kommunikationswege	**Lokal**	**Lokal**	**Verteilt**
Protokoll	Softbus	TCP/IP	TCP/IP
Kommunikations-Schnittstelle 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 möglich	nein	ja	ja
zwischen einer Instanz und einer realen Hardware-CPU	nein	nein	ja
zwischen einer Instanz und einer realen HMI V14	nein	nein	ja
zwischen einer Instanz und einer simulierten HMI V14	ja	ja	nein

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.

Kommunikations-Schnittstelle wählen

Die Kommunikations-Schnittstelle 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.

API-Funktionen zur Auswahl der Kommunikations-Schnittstelle

- GetCommunicationInterface() (Seite 122)
- SetCommunicationInterface() (Seite 122)
- CommunicationInterface { get; set; } (Seite 123)

Siehe auch

Schnittstellen - Informationen und Einstellungen (Seite 119)

S7-PLCSIM Advanced Control Panel (Seite 52)

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 (VMware) wie STEP 7 oder ein anderer Kommunikations-Partner.

Lokale Kommunikation über Softbus

Aus Sicherheitsgründen ist in PLCSIM Advanced die lokale Kommunikation über Softbus voreingestellt.

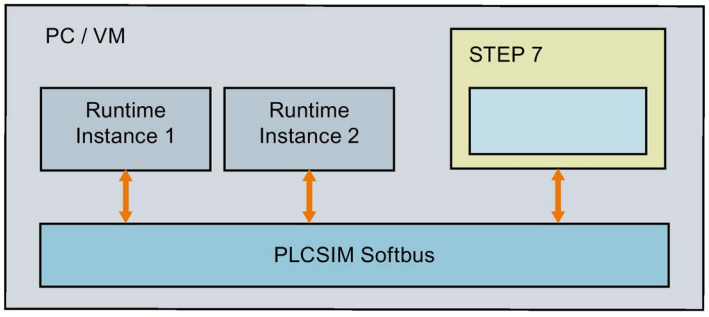


Bild 4-1 Lokale Kommunikation über Softbus

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.

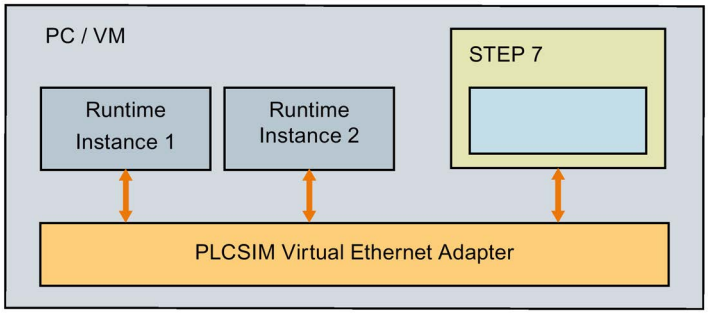


Bild 4-2 Lokale Kommunikation über TCP/IP

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. Die PCs sind über die Ethernet Adapter verbunden.

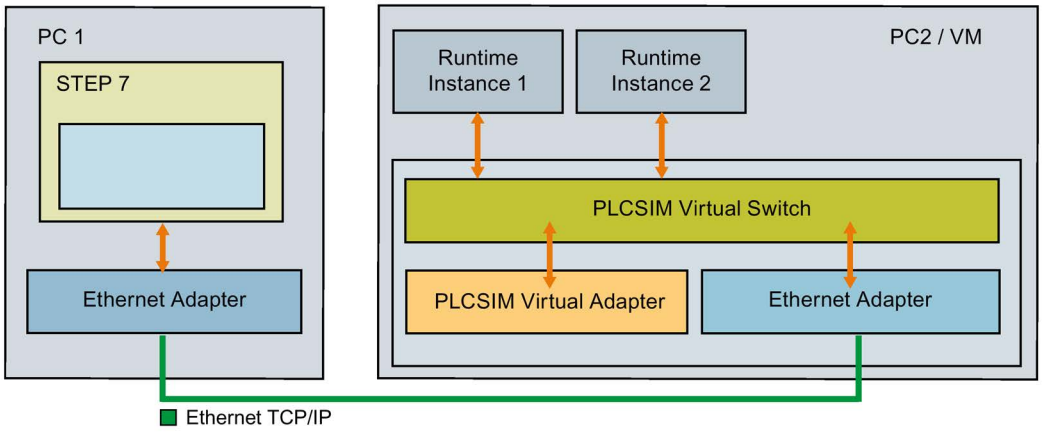


Bild 4-3 Verteilte Kommunikation über Ethernet

Beispiel 2: Verteilte Kommunikation

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 Netzwerkkarten verbunden.

Empfehlung

Verwenden Sie in den Einstellungen der VMware als Netzwerk Adapter Typ den Bridged Mode, um einen fehlerfreien Betrieb zu gewährleisten.

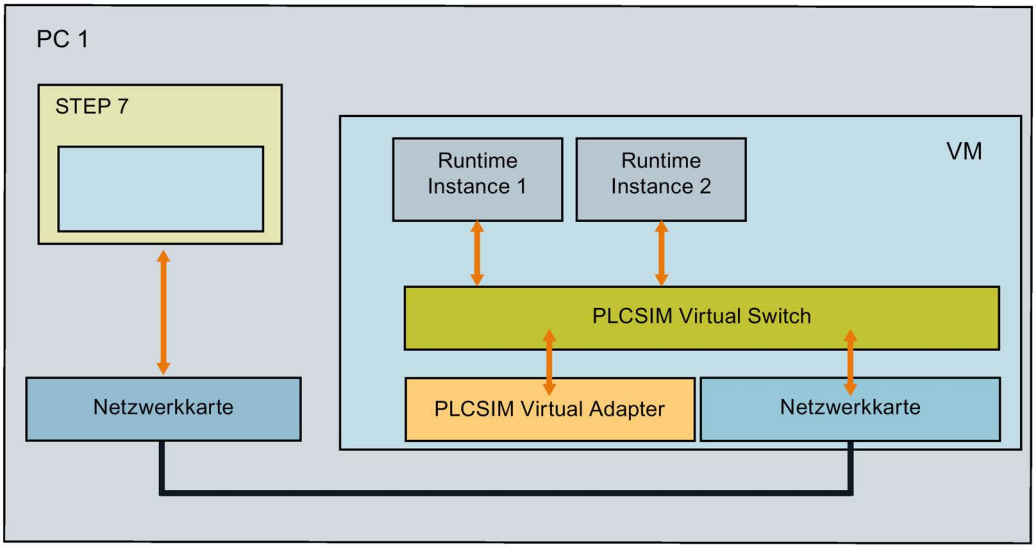


Bild 4-4 Verteilte Kommunikation über Netzwerkkarten

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 echten 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.

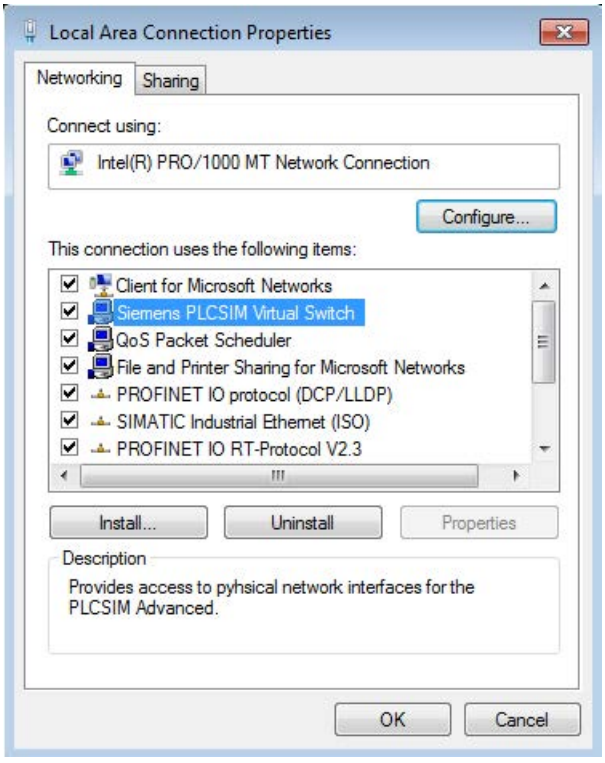


Bild 4-5 PLCSIM Virtual Switch aktivieren

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.

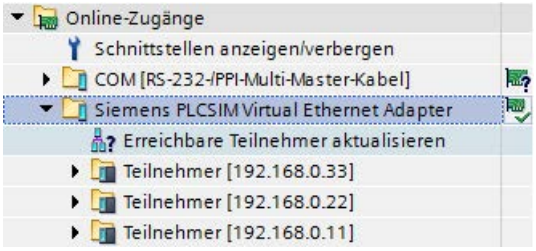


Bild 4-6 Erreichbare Teilnehmer am Virtual Ethernet Adapter

CPU simulieren

5

5.1 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 V14 und S7-PLCSIM Advanced V1.0 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".

Über das Control Panel Instanz erstellen und einschalten

- PLCSIM Advanced Control Panel öffnen
- 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
- Controller in RUN schalten, um Simulation zu starten
- Diagnose durchführen
- ...

Siehe auch

Simulations-Support (Seite 27)

5.2 Bedienoberfläche

5.2.1 S7-PLCSIM Advanced Symbol

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

Bild 5-1 PLCSIM Advanced Symbol

Nach einem Doppelklick auf das Symbol wird das Symbol im Infobereich der Taskleiste angezeigt.

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 die grafische Oberfläche von PLCSIM Advanced, das Control Panel.

Wenn das Control Panel geöffnet ist, können Sie die Mouse-over-Funktion nutzen, um Meldungen zum aktuellen Zustand der Instanzen anzuzeigen.

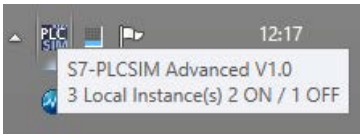
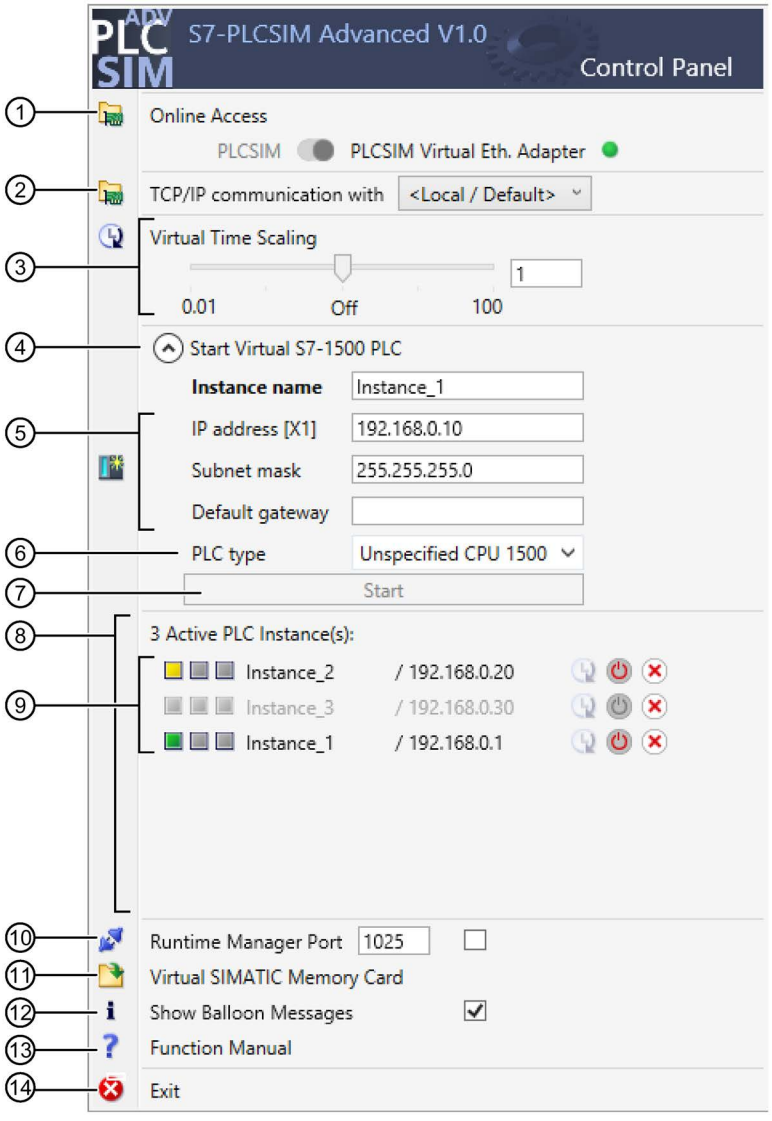


Bild 5-2 Beispiel: Meldung in der Taskleiste

5.2.2 S7-PLCSIM Advanced Control Panel

Control Panel

Das Control Panel ist optional und wird zum Betrieb von PLCSIM Advanced über die API nicht benötigt. Es steht in der Version 1.0 auf Englisch zur Verfügung.



①	Online Zugang	Schalter zur Auswahl der Kommunikations-Schnittstelle
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**	
	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.

⑤	• IP-Adresse • Subnetzmaske • Standard- Gateway	Die Eingabefelder werden sichtbar, wenn Sie die Kommunikations-Schnittstelle auf "PLCSIM Virtual Ethernet Adapter" umschalten.
⑥	CPU-Typ	Hier selektieren Sie den zu simulierenden CPU-Typ.
⑦	Schaltfläche "Start"	Mit der Schaltfläche erzeugen und starten Sie eine Instanz mit dem eingetragenen Namen.
⑧	Instanz-Liste	Die Liste zeigt die **lokal** vorhandenen Instanzen.
⑨	LED-Anzeigen	Die Bedeutung der LED wird angezeigt, wenn Sie den Mauszeiger darüber bewegen.
⑩	Runtime Manager Port	Hier öffnen Sie einen Port auf dem lokalen PC.
⑪	Virtual SIMATIC Memory Card	Hier öffnen Sie ein Explorer-Fenster mit dem Pfad zur virtuellen Speicherkarte.
⑫	Meldungen anzeigen	Hier deaktivieren Sie die PLCSIM Advanced Meldungen in der Windows Taskleiste für die Dauer der Bedienung.
⑬	Funktionshandbuch	Hier öffnen Sie das Funktionshandbuch S7-PLCSIM Advanced im Standard PDF-Viewer.
⑭	Exit	Exit schaltet alle Instanzen aus und schließt das Control Panel.

Bild 5-3 Control Panel

Schalter für Kommunikations-Schnittstelle

Mit dem Schalter wählen Sie für alle zu erzeugenden Instanzen die Kommunikations-Schnittstelle 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 Kommunikations-Schnittstelle, mit der eine Instanz startet, wird solange genutzt, bis alle Instanzen heruntergefahren werden.

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

Kommunikations-Schnittstelle wechseln

Um die Kommunikations-Schnittstelle 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 echte Netzwerkkarte auswählen. Sie aktivieren damit den PLCSIM Virtual Switch und stellen eine TCP/IP-Kommunikation zwischen den Instanzen und dem echten Netzwerk her.

Die Einstellung <Local > deaktiviert den PLCSIM Virtual Switch und trennt die Instanzen vom echten 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 Virtuelles Zeitverhalten (Seite 64).

Instanz erzeugen (lokal)

Um eine Instanz zu erzeugen, 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 Unspecified CPU 1500 oder Unspecified ET 200SP CPU aus. Mit dem ersten Download aus dem TIA Portal wird die CPU getauft.

Empfehlung: Erzeugen Sie maximal 4 lokale Instanzen auf Ihrem PC oder Ihrer Visualisierungsplattform. Remote-Instanzen können über das Control Panel nicht erzeugt werden.

Instanz-Liste

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

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,  Instanz abschalten,

 Instanz abschalten und vom Runtime Manager abmelden

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 den Link öffnen Sie das Verzeichnis.

Standardpfad: ...\Dokumente\Siemens\Simatic\Simulation\Runtime\Persistence

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

- Der Befehl schaltet alle lokalen Instanzen auf dem PC oder der VM aus und meldet sie vom Runtime Manager ab.
- 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.

Control Panel minimieren

Ein Klick auf eine freie Fläche auf dem Desktop minimiert das Control Panel. Die Instanzen bleiben davon unberührt.

5.3 Download

Voraussetzung

Sie können Sie 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.

Kommunikations-Schnittstelle 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

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.

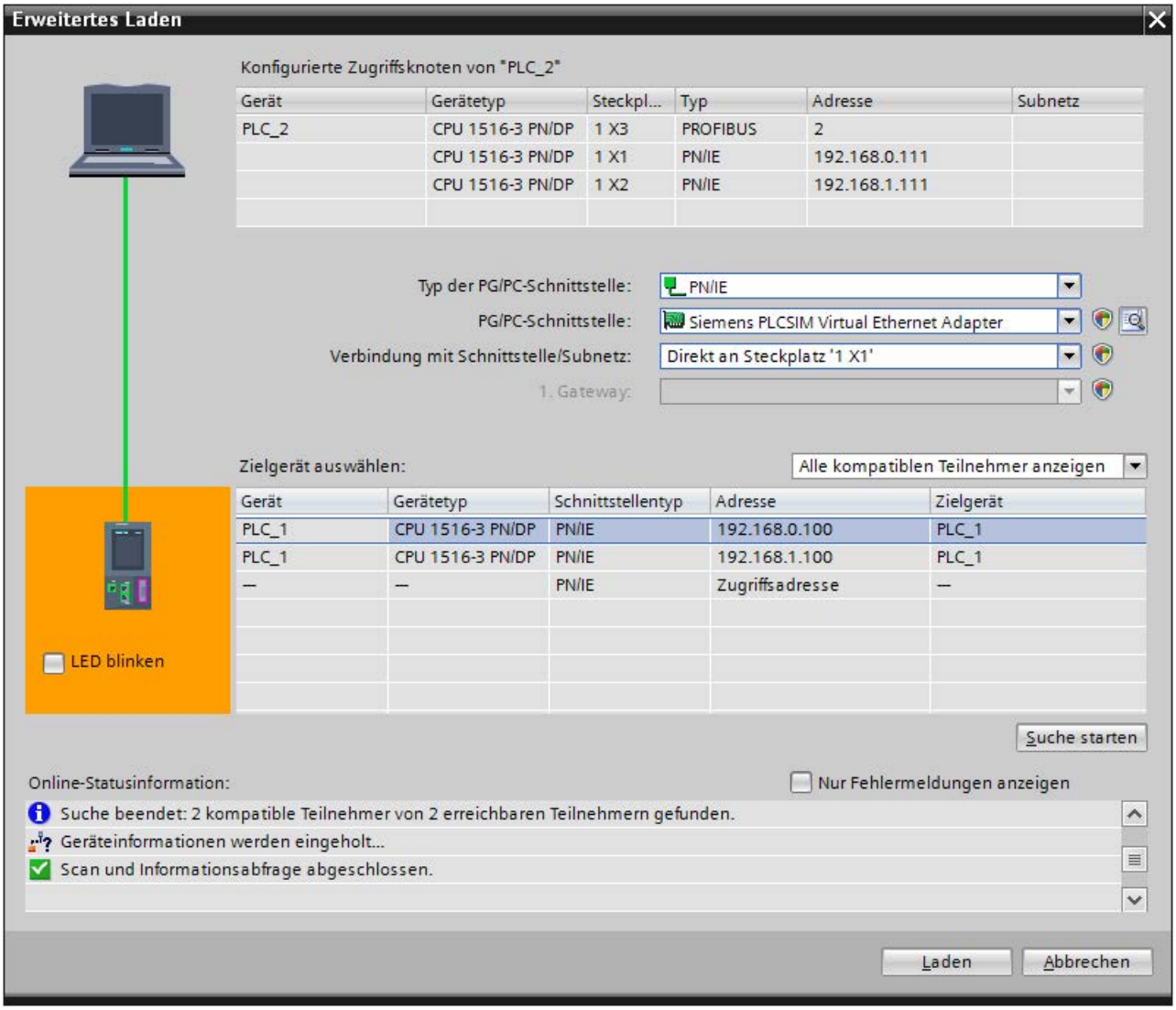


Bild 5-4 Beispiel: Download über den "PLCSIM Virtual Ethernet Adapter" (TCP/IP) nach der Taufe

5.4 MAC-Adresse der 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:

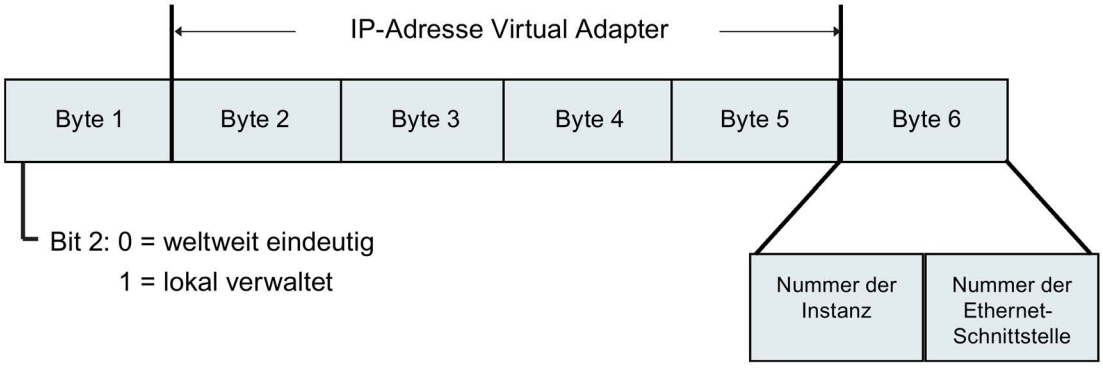


Bild 5-5 Aufbau der MAC-Adresse einer Instanz

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

Zuordnung der Ethernet-Schnittstellen

In PLCSIM Advanced V1.0 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.

5.5 Peripherie-I/O simulieren

Die Runtime API schreibt in einen und liest aus einem Speicherbereich. Dieser Speicher wird am Zykluskontrollpunkt 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 324)

5.6 Kommunikation simulieren

5.6.1 Simulierbare Kommunikationsdienste

S7-PLCSIM Advanced V1.0 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] • 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	• MB_CLIENT • MB_SERVER
E-Mail[2]	• TMAIL_C
S7-Kommunikation	• PUT / GET • BSEND / BRCV • USEND / URCV
OPC UA Server[2]	Datenaustausch mit OPC UA Clients
Webserver[2]	Datenaustausch über HTTP

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

[2] Nur über die Kommunikations-Schnittstelle "PLCSIM Virtual Ethernet Adapter" (TCP/IP)

Für die Kommunikation mit TUSEND / TURCV gelten Besonderheiten, siehe Einschränkungen bei Kommunikationsdiensten (Seite 319).

5.6.2 Kommunikation zwischen Instanzen

PLCSIM Advanced unterstützt die Kommunikation zwischen Instanzen. Eine Instanz kann eine Simulation in PLCSIM Advanced V1.0 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

S7-PLCSIM Advanced simuliert T-Bausteinverbindungen, für die das UDP-Protokoll konfiguriert ist, nur über die Kommunikations-Schnittstelle "PLCSIM Virtual Ethernet Adapter" (TCP/IP).

T-Bausteinanweisungen und Datensegmentierung

S7-PLCSIM Advanced implementiert T-Bausteinanweisungen mit einer Datensegmentierung von 4 kByte. Eine echte 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.

5.7 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.

Projektdaten offline bereitstellen

1. Legen Sie in STEP 7 in der Projektnavigation für die CPU unter 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 5-6 Card Reader hinzufügen

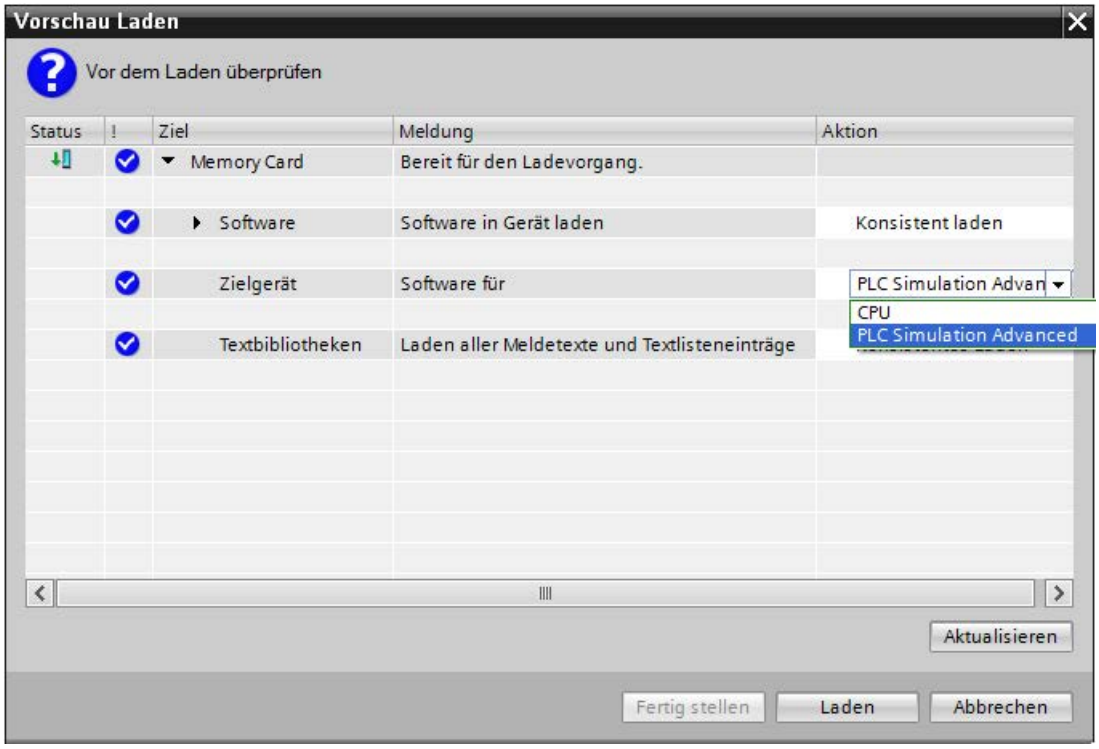


Bild 5-7 Vorschau Laden Dialog

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 130)
- StoragePath { get; set; } (Seite 131)

Siehe auch

Controller - Informationen und Einstellungen (Seite 125)

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.

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 228)
- SetSystemTime() (Seite 228)
- SystemTime { get; set; } (Seite 229)

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 229)
- SetScaleFactor() (Seite 230)
- ScaleFactor { get; set; } (Seite 231)

Siehe auch

Einstellungen für die virtuelle Zeit (Seite 228)

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.

Schnelllauf

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

Hinweis

CPU-Auslastung des PCs

Wenn der Skalierfaktor größer 1 ist, dann steigt die CPU-Auslastung des PCs, auf dem der virtuelle Controller simuliert wird, erheblich an.

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.

Zeitlupe

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

API-Funktionen

- GetScaleFactor() (Seite 229)
- SetScaleFactor() (Seite 230)
- ScaleFactor { get; set; } (Seite 231)

Siehe auch

Zykluskontrolle (Seite 232)

Einstellungen für die virtuelle Zeit (Seite 228)

Fehler bei Überlauf zyklischer Ereignisse (Seite 324)

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 Anwenderschnittstellen in einen Freeze-Zustand versetzt werden.

Ein ausgelöster Freeze-Zustand hält den virtuellen Controller am nächsten Zykluskontrollpunkt an. Es tritt folgendes ein:

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

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 `SROS_FREEZE` / `Freeze` an, siehe EOperatingState (Seite 302).

Freeze-Zustand auslösen

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

- **Zyklusgesteuerte Betriebsart** SingleStep und ExtendedSingleStep

  Siehe Simulations-Partner zyklusgesteuert synchronisieren (Seite 68).

- **Zeitgesteuerte Betriebsart** TimespanSynchronized

  Siehe Simulations-Partner zeitgesteuert synchronisieren (Seite 69).

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

API-Funktionen

- Einstellungen für die Zykluskontrolle (Seite 232)
- GetOperatingMode() (Seite 232)
- SetOperatingMode() (Seite 232)
- OperatingMode { get; set; } (Seite 233)

6.3 Simulations-Partner synchronisieren

6.3.1 Simulations-Partner zyklusgesteuert synchronisieren

Einleitung

Zur Synchronisation muss der virtuelle Controller in den Freeze-Zustand versetzt werden.

Eine Möglichkeit mehrere Simulations-Partner (Clients) zu synchronisieren, bieten die Betriebsarten SingleStep und ExtendedSingleStep.

Betriebsart SingleStep

In dieser Betriebsart geht der virtuelle Controller am Zykluskontrollpunkt in einen Freeze-Zustand und sendet ein Ereignis an die API-Clients.

Der Freeze-Zustand wird beendet, wenn ein API-Befehl den virtuellen Controller auffordert, den nächsten Schritt auszuführen oder in die Betriebsart Default zu wechseln.

API-Funktionen

- RunNextCycle() (Seite 236)

Betriebsart ExtendedSingleStep

In dieser Betriebsart geht der virtuelle Controller am Zykluskontrollpunkt in einen Freeze-Zustand und sendet ein Ereignis an die API-Clients.

Im Vergleich zur Betriebsart SingleStep überschreibt eine API-Funktion in dieser Betriebsart zusätzlich die Mindest-Zykluszeit des OB1-Zyklus. Wenn Sie eine Mindest-Zykluszeit von 200 ms definieren, dann ist der Mindestabstand zwischen zwei Zykluskontrollpunkten 200 virtuelle Millisekunden. Die Voreinstellung für die Betriebsart ist 100 ms.

Der Freeze-Zustand wird beendet, wenn ein API-Befehl den virtuellen Controller auffordert, den nächsten Schritt auszuführen oder in die Betriebsart Default zu wechseln.

API-Funktionen

- GetOverwrittenMinimalCycleTime_ns() (Seite 234)
- SetOverwrittenMinimalCycleTime_ns() (Seite 235)
- OverwrittenMinimalCycleTime_ns { get; set; } (Seite 235)
- RunNextCycle() (Seite 236)

Siehe auch

Zykluskontrolle (Seite 232)

6.3.2 Simulations-Partner zeitgesteuert synchronisieren

Einleitung

Eine Möglichkeit mehrere Simulations-Partner (Clients) zu synchronisieren, bietet die Betriebsart TimespanSynchronized des virtuellen Controllers.

Betriebsart TimespanSynchronized

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

Der Synchronisations-Master signalisiert einem Simulations-Client, dass er an der Reihe ist, 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 Zykluskontrollpunkt 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 den Clients 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.

API-Funktionen

- Einstellungen zur Zykluskontrolle (Seite 232)
- StartProcessing() (Seite 237)
- RunNextCycle() (Seite 236)

Anwenderschnittstellen (API) 7

7.1 Einführung

Komponenten der Simulation Runtime

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

- Runtime
  - Siemens.Simatic.Simulation.Runtime.Manager.exe
  - Siemens.Simatic.Simulation.Runtime.Instance.exe
- Bibliotheken
  - Siemens.Simatic.Simulation.Runtime.Api.x86.dll
  - Siemens.Simatic.Simulation.Runtime.Api.x64.dll
  - SimulationRuntimeApi.h
- Dokumentation der Schnittstellen mit Beispielen in Native C++ und .NET (Managed Code)

Beschreibung

Runtime und Bibliotheken	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	Headerdatei, die alle Datentypen beschreibt, die eine Native C++ Anwendung benötigt, um die API-Bibliothek zu verwenden.

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. weitere Simulations-Software oder eine GUI sein.

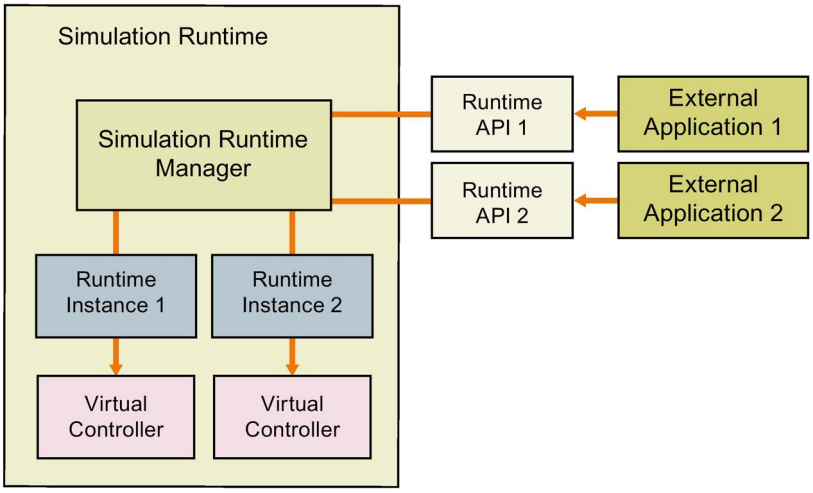
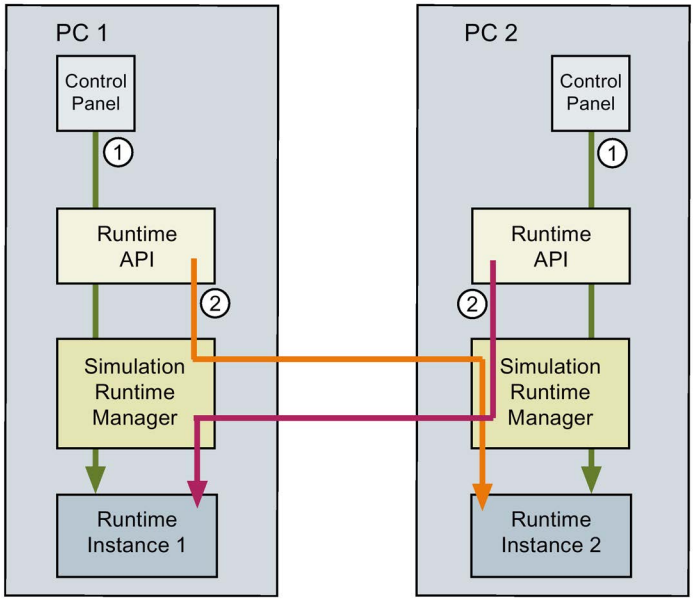


Bild 7-1 Externe Applikationen und Simulation Runtime

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 7-5 Übersicht IRemoteRuntimeManager Funktionen - Native C++ (Seite 78)
- Tabelle 7-9 Übersicht IRemoteRuntimeManager Funktionen - .NET (C#) (Seite 81)
- Tabelle 7-4 Übersicht IInstances Funktionen - Native C++ (Seite 76)
- Tabelle 7-8 Übersicht IInstances Funktionen - .NET (C#) (Seite 80)

Siehe auch

Übersicht Anwenderschnittstellen für Managed Code (Seite 79)

S7-PLCSIM Advanced Control Panel (Seite 52)

7.1.2 Anwenderschnittstellen (API)

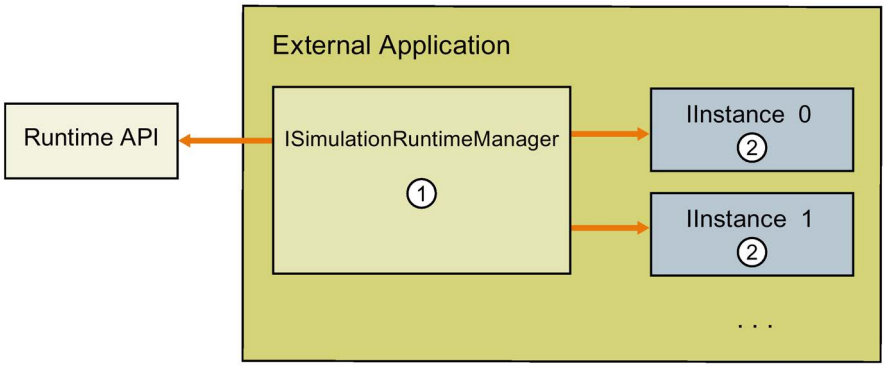
Die Anwenderschnittstellen der Simulation Runtime enthalten die Funktionen, mit denen Sie z. B. Instanzen erzeugen, den Betriebszustand eines virtuellen Controllers ändern und I/O-Daten austauschen.

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 74)
- Übersicht Datentypen für Native C++ (Seite 82)
- Übersicht Anwenderschnittstellen für Managed Code (Seite 79)
- Übersicht Datentypen für Managed Code (Seite 83)

Hinweis

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

7.1.3 Übersicht Anwenderschnittstellen für Native C++

API initialisieren und herunterfahren

Die folgende Tabelle zeigt, welche Funktionen für das Initialisieren und Herunterfahren der API für Native C++ vorhanden sind.

Tabelle 7- 1 Übersicht API initialisieren und herunterfahren - Native C++

Aktionen	Funktionen
API initialisieren	InitializeApi (Seite 84) RuntimeApiEntry_Initialize (Seite 85)
API herunterfahren (Seite 87)	`DestroyInterface` `RuntimeApiEntry_DestroyInterface`
API DLL abmelden (Seite 90)	`FreeApi` `ShutdownAndFreeApi`

Globale Funktionen

Tabelle 7- 2 Übersicht Globale Funktionen - Native C++

Aktionen	Funktionen

Globale Funktionen (Seite 92)	`GetNameOfAreaSection()`
	`GetNameOfCPUType()`
	`GetNameOfCommunicationInterface()`
	`GetNameOfDataType()`
	`GetNameOfLEDMode()`
	`GetNameOfLEDType()`
	`GetNameOfOperatingMode()`
	`GetNameOfOperatingState()`
	`GetNameOfPrimitiveDataType()`
	`GetNameOfTagListDetails()`
	`GetNameOfErrorCode()`
	`GetNameOfRuntimeConfigChanged()`
	`GetNameOfInstanceConfigChanged()`

API ISimulationRuntimeManager

Die folgende Tabelle zeigt, welche Funktionen für die API ISimulationRuntimeManager vorhanden sind.

Tabelle 7- 3 Übersicht API ISimulationRuntimeManager Funktionen - Native C++

Einstellungen und Informationen	Funktionen

Schnittstelle (Seite 96)	`GetVersion()`
	`IsInitialized()`
	`IsRuntimeManagerAvailable()`
	`Shutdown()`

Simulation Runtime Instanzen (Seite 99)	`GetRegisteredInstancesCount()`
	`GetRegisteredInstanceInfoAt()`
	`RegisterInstance()`
	`RegisterCustomInstance()`
	`CreateInterface()`

Remote-Verbindungen (Seite 107)	`OpenPort()`
	`ClosePort()`
	`GetPort()`
	`GetRemoteConnectionsCount()`
	`GetRemoteConnectionInfoAt()`
	`RemoteConnect()`

Ereignisse	

OnConfigurationChanged (Seite 112)	`RegisterOnConfigurationChangedCallback()`
	`UnregisterOnConfigurationChangedCallback()`
	`RegisterOnConfigurationChangedEvent()`
	`UnregisterOnConfigurationChangedEvent()`
	`WaitForOnConfigurationChangedEvent()`

OnRuntimeManagerLost (Seite 116)	`RegisterOnRuntimeManagerLostCallback()`
	`UnregisterOnRuntimeManagerLostCallback()`
	`RegisterOnRuntimeManagerLostEvent()`
	`UnregisterOnRuntimeManagerLostEvent()`
	`WaitForOnRuntimeManagerLostEvent()`

API IInstances

Die folgende Tabelle zeigt, welche Funktionen für die API IInstances vorhanden sind.

Tabelle 7- 4 Übersicht IInstances Funktionen - Native C++

Einstellungen und Informationen	Funktionen

Schnittstelle (Seite 119)	`GetID()`
	`GetName()`
	`GetCPUType()`
	`SetCPUType()`
	`GetCommunicationInterface()`
	`SetCommunicationInterface()`
	`GetInfo()`
	`UnregisterInstance()`

Controller (Seite 125)	`GetControllerName()`
	`GetControllerShortDesignation()`
	`GetControllerIPCount()`
	`GetControllerIP()`
	`GetControllerIPSuite4()`
	`SetIPSuite()`
	`GetStoragePath()`
	`SetStoragePath()`
	`ArchiveStorage()`
	`RetrieveStorage()`

Betriebszustand (Seite 134)	`GetOperatingState()`
	`PowerOn()`
	`PowerOff()`
	`MemoryReset()`
	`Run()`
	`Stop()`

Variablentabelle (Seite 145)	`UpdateTagList()`
	`GetTagListStatus()`
	`GetTagInfoCount()`
	`GetTagInfos()`
	`CreateConfigurationFile()`

I/O-Zugriff über Adresse - Lesen (Seite 151)	`GetAreaSize()`
	`ReadBit()`
	`ReadByte()`
	`ReadBytes()`
	`ReadSignals()`

I/O-Zugriff über Adresse - Schreiben (Seite 160)	`WriteBit()`
	`WriteByte()`
	`WriteBytes()`
	`WriteSignals()`

I/O-Zugriff über Variablenname - Lesen (Seite 168)	`Read()`
	`ReadBool()`
	`ReadChar(), ReadWChar()`
	`ReadDouble()`
	`ReadFloat()`
	`ReadInt8(), ReadInt16(), ReadInt32(), ReadInt64()`
	`ReadUInt8(), ReadUInt16(), ReadUInt32(), ReadUInt64()`
	`ReadSignals()`

I/O-Zugriff über Variablenname - Schreiben (Seite 198)	`Write()`
	`WriteBool()`
	`WriteChar(), WriteWChar()`
	`WriteDouble()`
	`WriteFloat()`
	`WriteInt8(), WriteInt16(), WriteInt32(), WriteInt64(),`
	`WriteUInt8(),WriteUInt16(), WriteUInt32(), WriteUInt64()`
	`WriteSignals()`

Einstellungen und Informationen	**Funktionen**

Virtuelle Zeit (Seite 228)	`GetSystemTime()`
	`SetSystemTime()`
	`GetScaleFactor()`
	`SetScaleFactor()`

Zykluskontrolle (Seite 232)	`GetOperatingMode()`
	`SetOperatingMode()`
	`SetAlwaysSendOnEndOfCycleEnabled()`
	`IsAlwaysSendOnEndOfCycleEnabled()`
	`GetOverwrittenMinimalCycleTime_ns()`
	`SetOverwrittenMinimalCycleTime_ns()`
	`RunNextCycle()`
	`StartProcessing()`

Ereignisse	

OnOperatingStateChanged (Seite 239)	`RegisterOnOperatingStateChangedCallback()`
	`UnregisterOnOperatingStateChangedCallback()`
	`RegisterOnOperatingStateChangedEvent()`
	`UnregisterOnOperatingStateChangedEvent()`
	`WaitForOnOperatingStateChangedEvent()`

OnEndOfCycle (Seite 244)	`RegisterOnEndOfCycleCallback()`
	`UnregisterOnEndOfCycleCallback()`
	`RegisterOnEndOfCycleEvent()`
	`UnregisterOnEndOfCycleEvent()`
	`WaitForOnEndOfCycleEvent()`

OnConfigurationChanging (Seite 247)	`RegisterOnConfigurationChangingCallback()`
	`UnregisterOnConfigurationChangingCallback()`
	`RegisterOnConfigurationChangingEvent()`
	`UnregisterOnConfigurationChangingEvent()`
	`WaitForOnConfigurationChangingEvent()`

OnConfigurationChanged (Seite 250)	`RegisterOnConfigurationChangedCallback()`
	`UnregisterOnConfigurationChangedCallback()`
	`RegisterOnConfigurationChangedEvent()`
	`UnregisterOnConfigurationChangedEvent()`
	`WaitForOnConfigurationChangedEvent()`

OnLedChanged (Seite 253)	`RegisterOnLedChangedCallback()`
	`UnregisterOnLedChangedCallback()`
	`RegisterOnLedChangedEvent()`
	`UnregisterOnLedChangedEvent()`
	`WaitForOnLedChangedEvent()`

API IRemoteRuntimeManager

Die folgende Tabelle zeigt, welche Funktionen für die API IRemoteRuntimeManager (Remote-Verbindungen) vorhanden sind:

Tabelle 7- 5 Übersicht IRemoteRuntimeManager Funktionen - Native C++

Einstellungen und Informationen	**Funktionen**

Schnittstelle (Seite 256)	`GetVersion()`
	`GetIP()`
	`GetPort()`
	`GetRemoteComputerName()`
	`Disconnect()`

Simulation Runtime Instanzen (Seite 260)	`GetRegisteredInstancesCount()`
	`GetRegisteredInstanceInfoAt()`
	`RegisterInstance()`
	`RegisterCustomInstance()`
	`CreateInterface()`

Ereignisse	

OnConnectionLost (Seite 268)	`RegisterOnConnectionLostCallback()`
	`UnregisterOnConnectionLostCallback()`
	`RegisterOnConnectionLostEvent()`
	`UnregisterOnConnectionLostEvent()`
	`WaitForOnConnectionLostEvent()`

7.1.4 Übersicht Anwenderschnittstellen für Managed Code

API initialisieren und herunterfahren

Tabelle 7- 6 Übersicht API initialisieren und herunterfahren - .NET (C#)

Aktionen	Funktionen
API initialisieren (Seite 87)	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeManager`
API herunterfahren (Seite 92)	

API ISimulationRuntimeManager

Die folgende Tabelle zeigt, welche Funktionen für die API ISimulationRuntimeManager vorhanden sind.

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

Einstellungen und Informationen	Funktionen

Schnittstelle (Seite 96)	`Version { get; }`
	`IsInitialized { get; }`
	`IsRuntimeManagerAvailable { get; }`
	`Shutdown()`

Simulation Runtime Instanzen (Seite 99)	`RegisterInstanceInfo { get; }`
	`RegisterInstance()`
	`RegisterCustomInstance()`
	`CreateInterface()`

Remote Verbindungen (Seite 107)	`OpenPort()`
	`ClosePort()`
	`Port { get; }`
	`RemoteConnectionInfo { get; }`
	`RemoteConnect()`

Ereignisse	

OnConfigurationChanged (Seite 112)	`OnConfigurationChanged`
	`RegisterOnConfigurationChangedEvent()`
	`UnregisterOnConfigurationChangedEvent()`
	`WaitForOnConfigurationChangedEvent()`

OnRuntimeManagerLost (Seite 116)	`OnRuntimeManagerLost()`
	`RegisterOnRuntimeManagerLostEvent()`
	`UnregisterOnRuntimeManagerLostEvent()`
	`WaitForOnRuntimeManagerLostEvent()`

API IInstances

Die folgende Tabelle zeigt, welche Funktionen für die API IInstances vorhanden sind.

Tabelle 7- 8 Übersicht IInstances Funktionen - .NET (C#)

Einstellungen und Informationen	Funktionen

Schnittstelle (Seite 119)	`Dispose ()`
	`ID { get; }`
	`Name { get; }`
	`CPUType { get; set; }`
	`CommunicationInterface { get; }`
	`Info { get; }`
	`UnregisterInstance()`

Controller - Informationen und Einstellungen (Seite 125)	`ControllerName { get; }`
	`ControllerShortDesignation { get; }`
	`ControllerIPSuite4 { get; }`
	`SetIPSuite()`
	`StoragePath { get; set; }`
	`ArchiveStorage()`
	`RetrieveStorage()`

Betriebszustand (Seite 134)	`OperatingState { get; }`
	`PowerOn()`
	`PowerOff()`
	`MemoryReset()`
	`Run()`
	`Stop()`

Variablentabelle (Seite 145)	`UpdateTagList()`
	`GetTagListStatus()`
	`TagInfos { get; }`
	`CreateConfigurationFile()`

I/O-Zugriff über Adresse - Lesen (Seite 151)	`InputArea | MarkerArea | OutputArea { get; }`
	`AreaSize { get; }`
	`ReadBit()`
	`ReadByte()`
	`ReadBytes()`
	`ReadSignals()`

I/O-Zugriff über Adresse - Schreiben (Seite 160)	`WriteBit()`
	`WriteByte()`
	`WriteBytes()`
	`WriteSignals()`

I/O-Zugriff über Variablenname - Lesen (Seite 168)	`Read()`
	`ReadBool()`
	`ReadChar(), ReadWChar()`
	`ReadDouble()`
	`ReadFloat()`
	`ReadInt8(), ReadInt16(), ReadInt32(), ReadInt64()`
	`ReadUInt8(), ReadUInt16(), ReadUInt32(), ReadUInt64()`
	`ReadSignals()`

I/O-Zugriff über Variablenname - Schreiben (Seite 198)	`Write()`
	`WriteBool()`
	`WriteChar(), WriteWChar()`
	`WriteDouble()`
	`WriteFloat()`
	`WriteInt8(), WriteInt16(), WriteInt32(), WriteInt64(),`
	`WriteUInt8(),WriteUInt16(), WriteUInt32(), WriteUInt64()`
	`WriteSignals()`

Virtuelle Zeit (Seite 228)	`SystemTime { get; set; }` `ScaleFactor { get; set; }`

Einstellungen und Informationen	Funktionen

Zykluskontrolle (Seite 232)	`OperatingMode { get; set; }`
	`IsAlwaysSendOnEndOfCycleEnabled { get; set; }`
	`OverwrittenMinimalCycleTime_ns { get; set; }`
	`RunNextCycle()`
	`StartProcessing()`

Ereignisse	

OnOperatingStateChanged (Seite 239)	`OnOperatingStateChanged`
	`RegisterOnOperatingStateChangedEvent()`
	`UnregisterOnOperatingStateChangedEvent()`
	`WaitForOnOperatingStateChangedEvent()`

OnEndOfCycle (Seite 244)	`OnEndOfCycle`
	`RegisterOnEndOfCycleEvent()`
	`UnregisterOnEndOfCycleEvent()`
	`WaitForOnEndOfCycleEvent()`

OnConfigurationChanging (Seite 247)	`OnConfigurationChanging`
	`RegisterOnConfigurationChangingEvent()`
	`UnregisterOnConfigurationChangingEvent()`
	`WaitForOnConfigurationChangingEvent()`

OnConfigurationChanged (Seite 112)	`OnConfigurationChanged`
	`RegisterOnConfigurationChangedEvent()`
	`UnregisterOnConfigurationChangedEvent()`
	`WaitForOnConfigurationChangedEvent()`

OnLedChanged (Seite 253)	`OnLedChanged`
	`RegisterOnLedChangedEvent()`
	`UnregisterOnLedChangedEvent()`
	`WaitForOnLedChangedEvent()`

API IRemoteRuntimeManager

Die folgende Tabelle zeigt, welche Funktionen für die API IRemoteRuntimeManager vorhanden sind.

Tabelle 7- 9 Übersicht IRemoteRuntimeManager Funktionen - .NET (C#)

Einstellungen und Informationen	Funktionen

Schnittstelle (Seite 256)	`Dispose()`
	`Version { get; }`
	`IP { get; }`
	`Port { get; }`
	`RemoteComputerName { get; }`
	`Disconnect()`

Simulation Runtime Instanzen (Seite 99)	`RegisterInstanceInfo { get; }`
	`RegisterInstance()`
	`RegisterCustomInstance()`
	`CreateInterface()`

Ereignisse	

OnConnectionLost() (Seite 268)	`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- 10 Übersicht Datentypen - Native C++

Datentyp	
DLL-Importfunktionen (Seite 273)	`ApiEntry_Initialize` `ApiEntry_DestroyInterface`

Event Callback- Funktionen (Seite 274)	`EventCallback_VOID`
	`EventCallback_II_SREC_ST`
	`EventCallback_II_SREC_ST_SROS_SROS`
	`EventCallback_II_SREC_ST_SRLT_SRLM`
	`EventCallback_II_SREC_ST_INT64_UINT32`
	`EventCallback_IRRTM`
	`EventCallback_SRCC_UINT32_UINT32_INT32`
	`EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32`

Definitionen und Konstanten (Seite 286)	
Unions (Seite 287)	`UIP` `UDataValue`

Strukturen (Seite 289)	`SDataValue`
	`SDataValueByAddress`
	`SDataValueByName`
	`SConnectionInfo`
	`SInstanceInfo`
	`SDimension`
	`STagInfo`
	`SIP`
	`SIPSuite4`

Aufzählungen (Seite 298)	`ERuntimeErrorCode`
	`EArea`
	`EOperatingState`
	`EOperatingMode`
	`ECPUType`
	`ECommunicationInterface`
	`ELEDType`
	`ELEDMode`
	`EPrimitiveDataTypes`
	`EDataType`
	`ETagListDetails`
	`ERuntimeConfigChanged`
	`EInstanceConfigChanged`

7.1.6 Übersicht Datentypen für Managed Code

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

Tabelle 7- 11 Übersicht Datentypen - .NET (C#)

Datentyp	

Delegat Definitionen (Seite 280)
- Event Handler Methoden
	`Delegate_Void`
	`Delegate_II_EREC_DT`
	`Delegate_II_EREC_DT_EOS_EOS`
	`Delegate_II_EREC_DT_ELT_ELM`
	`Delegate_II_EREC_DT_INT64_UINT32`
	`Delegate_IRRTM`
	`Delegate_SRCC_UINT32_UINT32_INT32`
	`Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32`

Definitionen und Konstanten (Seite 286)	

Strukturen (Seite 289)	`SDataValue`
	`SDataValueByAddress`
	`SDataValueByName`
	`SConnectionInfo`
	`SInstanceInfo`
	`SDimension`
	`STagInfo`
	`SIP`
	`SIPSuite4`

Aufzählungen (Seite 298)	`ERuntimeErrorCode`
	`EArea`
	`EOperatingState`
	`EOperatingMode`
	`ECPUType`
	`ECommunicationInterface`
	`ELEDType`
	`ELEDMode`
	`EPrimitiveDataTypes`
	`EDataType`
	`ETagListDetails`
	`ERuntimeConfigChanged`
	`EInstanceConfigChanged`

7.2 API initialisieren

7.2.1 Native C++

7.2.1.1 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. Die DLL wird aus dem Startup-Verzeichnis jener Anwendung geladen, die diese Funktion aufruft oder vom Verzeichnis, das die Pfad Parameter liefern.

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- 12 InitializeApi() - Native C++

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

Parameter	• `ISimulationRuntimeManager**` `out_SimulationRuntimeManagerInterface:` Zeiger auf einen Runtime Manager Schnittstellenzeiger. Der Zeiger muss mit `NULL` initialsiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt. Siehe Datentypen (Seite 272). • `WCHAR* in_SimulationRuntimeApiDllPath`: Der Pfad zur Runtime API-Bibliothek (DLL).	
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 zu der Version, mit der die API kompiliert wurde.
	`SREC_CONNECTION_ERROR`	Zum Runtime Manager kann keine Verbindung hergestellt werden.
	`SREC_ERROR_LOADING_DLL`	Die API-Bibliothek (DLL) kann nicht geladen werden.

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 88).

7.2.1.2 RuntimeApiEntry_Initialize

Beschreibung

Nutzen Sie die Funktion `RuntimeApiEntry_Initialize` nur, wenn die API-Bibliothek (DLL) von einem anderen Verzeichnis geladen werden soll als dem Startup-Verzeichnis jener Anwendung, 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- 13 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` initialsiert werden. Die Schnittstelle wird innerhalb der Funktion erzeugt. Siehe Datentypen (Seite 272). • `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 zu der Version, mit der die API kompiliert wurde.
	SREC_CONNECTION_ERROR	Zum Runtime Manager kann keine Verbindung hergestellt werden.
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 88).

7.2.2 .NET (C#)

7.2.2.1 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- 14 Initialize - .NET (C#)

Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationInitializationException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.ConnectionError`	Zum Runtime Manager kann keine Verbindung hergestellt werden.

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 90)
- ShutdownAndFreeApi() (Seite 91)

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- 15 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 88).

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

Tabelle 7- 16 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- 17 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- 18 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 auf folgende Arten beenden:

- Durch Bereinigung über den .NET Garbage Collector.
- Für die IInstance und IRemoteRuntimeManager Schnittstellen über den Aufruf der Funktion Dispose (Seite 119).

API manuell bereinigen

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

1. Löschen Sie alle Schnittstellen. Schnittstellen - Informationen und Einstellungen (Seite 119)
2. Rufen Sie die Funktion Shutdown() (Seite 96) der ISimulationRuntimeManager Schnittstelle auf.

7.4 Globale Funktionen (Native C++)

GetNameOfAreaSection()

Liefert den Namen des Aufzählungseintrags zurück.

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

Syntax	`const WCHAR* GetNameOfAreaSection( EArea in_AreaSection );`
Parameter	• `EArea in_AreaSection:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfCPUType()

Liefert den Namen des Aufzählungseintrags zurück.

Tabelle 7- 20 GetNameOfCPUType() - Native C++

Syntax	`const WCHAR* GetNameOfCPUType( ECPUType in_CPUType );`
Parameter	• `ECPUType in_CPUType:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfCommunicationInterface()

Liefert den Namen des Aufzählungseintrags zurück.

Tabelle 7- 21 GetNameOfCommunicationInterface() - Native C++

Syntax	`const WCHAR* GetNameOfCommunicationInterface(`
	`  ECommunicationInterface in_CommunicationInterface`
	`);`

Parameter	• `ECommunicationInterface in_CommunicationInterface:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfDataType()

Liefert den Namen des Aufzählungseintrags zurück.

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

Syntax	`const WCHAR* GetNameOfDataType(`
	`  EDataType in_DataType`
	`);`

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

GetNameOfErrorCode()

Liefert den Namen des Aufzählungseintrags zurück.

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

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

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

GetNameOfLEDMode()

Liefert den Namen des Aufzählungseintrags zurück.

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

Syntax	`const WCHAR* GetNameOfLEDMode(`
	`  ELEDMode in_LEDMode`
	`);`

Parameter	• `ELEDMode in_LEDMode:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfLEDType()

Liefert den Namen des Aufzählungseintrags zurück.

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

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

GetNameOfOperatingMode()

Liefert den Namen des Aufzählungseintrags zurück.

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

Syntax	`const WCHAR* GetNameOfOperatingMode( EOperatingMode in_OperatingMode );`
Parameter	• `EOperatingMode in_OperatingMode:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfErrorCode()

Liefert den Namen des Aufzählungseintrags zurück.

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

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

GetNameOfOperatingState

Liefert den Namen des Aufzählungseintrags zurück.

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

Syntax	`const WCHAR* GetNameOfOperatingState( EOperatingState in_OperatingState );`
Parameter	• `EOperatingState in_OperatingState:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfPrimitiveDataType

Liefert den Namen des Aufzählungseintrags zurück.

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

Syntax	`const WCHAR* GetNameOfPrimitiveDataType( EPrimitiveDataType in_DataType );`
Parameter	• `EPrimitiveDataType in_DataType:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfTagListDetails

Liefert den Namen des Aufzählungseintrags zurück.

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

Syntax	`const WCHAR* GetNameOfTagListDetails( ETagListDetails in_TagListDetails );`
Parameter	• `ETagListDetails in_TagListDetails:` Aufzählungseintrag
Rückgabewerte	`const WCHAR*:` Name des Aufzählungseintrags

GetNameOfRuntimeConfigChanged()

Liefert den Namen des Aufzählungseintrags zurück.

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

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

GetNameOfInstanceConfigChanged()

Liefert den Namen des Aufzählungseintrags zurück.

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

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

Siehe auch

EPrimitiveDataType (Seite 308)

EDataType (Seite 310)

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- 33 GetVersion() - Native C++

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

Tabelle 7- 34 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- 35 IsInitialized() - Native C++

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

Tabelle 7- 36 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 112).

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

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

Tabelle 7- 38 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- 39 Shutdown() - Native C++

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

Tabelle 7- 40 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- 41 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- 42 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 Datentypen (Seite 286).	
Rückgabewerte	Runtime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`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 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- 43 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- 44 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 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 286). • `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 88).

Tabelle 7- 45 RegisterInstance() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2">
<pre>IInstance RegisterInstance();
IInstance RegisterInstance(
 string in_InstanceName
);
IInstance RegisterInstance(
 ECPUType in_CPUType
);
IInstance RegisterInstance(
 ECPUType in_CPUType
 string in_InstanceName
);</pre>
</td></tr>
<tr><td>Parameter</td><td colspan="2">

<code>ECPUType in_CPUType:</code>

Definiert, welcher CPU-Typ beim Start der Instanz simuliert wird. Die Voreinstellung ist <code>"ECPUType.Unspecified"</code>.

Wenn über STEP 7 oder von der Virtual Memory Card ein anderer CPU-Typ geladen wird, dann gilt dieser CPU-Typ.
<code>string in_InstanceName:</code>

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 <code>"Instance_#"</code> (# ist die ID der Instanz). Wenn dieser Name bereits existiert, wird der Name <code>"Instance_#.#"</code> verwendet, wobei das zweite # ein Zähler ist, der solange erhöht wird, bis der Name eindeutig ist. Die Länge des Namens muss kürzer sein als <code>DINSTANCE_NAME_LENGTH</code>. Siehe Datentypen (Seite 286).

</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers, ansonsten einen Null-Zeiger.</td></tr>
<tr><td>Ausnahmen</td><td colspan="2"><code>Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException</code></td></tr>
<tr><td></td><td>Runtime Fehlercode</td><td>Bedingung</td></tr>
<tr><td></td><td><code>ERuntimeErrorCode.Timeout</code></td><td>Die Funktion kehrt nicht rechtzeitig wieder.</td></tr>
<tr><td></td><td><code>ERuntimeErrorCode.WrongArgument</code></td><td>Der Name ist ungültig.</td></tr>
<tr><td></td><td><code>ERuntimeErrorCode.LimitReached</code></td><td>Es sind bereits 16 Instanzen im Runtime Manager registriert.</td></tr>
<tr><td></td><td><code>ERuntimeErrorCode.AlreadyExists</code></td><td>Eine Instanz mit diesem Namen existiert bereits.</td></tr>
</table>

RegisterCustomInstance()

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

Tabelle 7- 46 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 286). • `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("C:\\Temp\\vplc.dll");`
	`}`	

Hinweis

Native C++

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

Siehe DestroyInterface() (Seite 88).

Tabelle 7- 47 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 286).	
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- 48 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 88).

Tabelle 7- 49 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- 50 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- 51 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.
	`ERuntimeError-` `Code.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- 52 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- 53 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- 54 GetPort() - Native C++

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

Tabelle 7- 55 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- 56 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- 57 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- 58 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- 59 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` initialsiert 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.

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 88).

Tabelle 7- 60 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
	`ERuntimeError-Code.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.

Siehe auch

Datentypen (Seite 272)

Variablentabelle (Seite 145)

7.5.4 Ereignisse

Ereignisse für die Schnittstelle ISimulationRuntimeManager

In der API werden folgende Ereignisse unterschieden:

Tabelle 7- 61 Ereignisse für die Schnittstelle ISimulationRuntimeManager

Ereignis	Ursache
OnConfigurationChanged (Seite 112)	Dieses Ereignis wird ausgelöst bei einer Änderung der Runtime Manager Konfiguration. Z. B. wenn eine neue Instanz registriert, eine Instanz entfernt oder eine Verbindung zu einem Client aufgebaut wird. Das Control Panel nutzt ein solches Ereignis, um die Liste der verfügbaren Instanzen zu aktualisieren.
OnRuntimeManagerLost (Seite 116)	Dieses Ereignis wird ausgelöst, wenn die Verbindung zum Runtime Manager unterbrochen ist.

7.5.4.1 OnConfigurationChanged

OnConfigurationChanged

Meldet eine Event-Handler Methode an oder ab.

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

Syntax	`event Delegate_SRCC_UINT32_UINT32_INT32 OnConfigurationChanged;`
Parameter	Keine. Siehe Delegate_SRCC_UINT32_UINT32_INT32 (Seite 284).
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- 63 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 278).
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- 64 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- 65 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- 66 UnregisterOnConfigurationChangedCallback() - Native C++

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

UnregisterOnConfigurationChangedEvent()

Meldet das Event-Objekt ab.

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

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

Tabelle 7- 68 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- 69 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- 70 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 OnRuntimeManagerLost

OnRuntimeManagerLost

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 71 OnRuntimeManagerLost - .NET (C#)

Syntax	`event Delegate_Void OnRuntimeManagerLost;`
Parameter	Keine. Siehe Delegate_Void (Seite 280).
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- 72 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 274).
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- 73 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- 74 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- 75 UnregisterOnRuntimeManagerLostCallback() - Native C++

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

UnregisterOnRuntimeManagerLostEvent()

Meldet das Event-Objekt ab.

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

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

Tabelle 7- 77 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- 78 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- 79 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.

7.6 API IInstances

7.6.1 Schnittstellen - Informationen und Einstellungen

Dispose()

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

Tabelle 7- 80 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- 81 GetID() - Native C++

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

Tabelle 7- 82 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- 83 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 286). • `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- 84 Name { get; } - .NET (C#)

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

GetCPUType()

Liefert den CPU-Typ des virtuellen Controllers zurück.

Tabelle 7- 85 GetCPUType() - Native C++

Syntax	`ECPUType GetCPUType();`
Parameter	Keine
Rückgabewerte	Ein Aufzählungselement, das den CPU-Typ definiert. Siehe ECPUType (Seite 303).

SetCPUType()

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

Tabelle 7- 86 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- 87 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 Kommunikations-Schnittstelle des virtuellen Controllers zurück: Lokale Kommunikation (Softbus) oder TCPIP. Ein Wechsel der Kommunikations-Schnittstelle erfolgt nur bei einem Neustart des Controllers. Alle Instanzen, die gestartet werden, müssen die selbe Kommunikations-Schnittstelle nutzen.

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

Tabelle 7- 88 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 Kommunikations-Schnittstelle des virtuellen Controllers: Lokale Kommunikation (Softbus) oder TCPIP. Ein Wechsel der Kommunikations-Schnittstelle erfolgt nur bei einem Neustart des Controllers. Alle Instanzen, die gestartet werden, müssen die selbe Kommunikations-Schnittstelle nutzen.

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

Tabelle 7- 89 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 Kommunikations-Schnittstelle des virtuellen Controllers oder liefert sie zurück: Lokale Kommunikation (Softbus) oder TCPIP. Ein Wechsel der Kommunikations-Schnittstelle erfolgt nur beim Neustart des Controllers. Alle Instanzen, die gestartet werden, müssen die selbe Kommunikations-Schnittstelle nutzen.

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

Tabelle 7- 90 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- 91 GetInfo() - Native C++

Syntax	`SInstanceInfo GetInfo();`
Parameter	Keine
Rückgabewerte	`SInstanceInfo`: Eine Struktur, die Informationen über die Instanz bereitstellt. Siehe SInstanceInfo (Seite 292).

Tabelle 7- 92 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- 93 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- 94 UnregisterInstance() - .NET (C#)

Syntax	`void UnregisterInstance();`	
Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.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- 95 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- 96 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
	`ERuntimeError-` `Code.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- 97 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- 98 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
	`ERuntimeError-` `Code.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- 99 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- 100 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- 101 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 Kommunikations-Schnittstelle "Softbus" genutzt wird, sind Subnetzmaske und Standard-Gateway 0.

Tabelle 7- 102 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- 103 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- 104 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 Kommunikations-Schnittstelle "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 Kommunikations-Schnittstelle "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- 105 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 Kommunikations-Schnittstelle "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 Kommunikations-Schnittstelle "Softbus" ist, dann wird dieses Flag ignoriert.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.DoesNotExist`	Es gibt keine Netzwerk-Schnittstelle mit dieser ID.
	`ERuntimeError-Code.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- 106 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 272). • `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- 107 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 272).	
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 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- 108 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.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- 109 ArchiveStorage() - Native C++

Syntax	`ERuntimeErrorCode ArchiveStorage(`
	`  WCHAR* in_FullFileName`
	`);`	

Parameter	• `WCHAR in_FullFileName:` Der vollständige Pfad zur ZIP-Datei.	
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.

Tabelle 7- 110 ArchiveStorage() - .NET (C#)

Syntax	`void ArchiveStorage(`
	`  string in_ArchiveStorageFile`
	`);`	

Parameter	• `string in_ArchiveStorageFile:` Der vollständige Pfad zur ZIP-Datei. Der Pfad bezieht sich auf Verzeichnisse des Computers, auf dem die Instanz läuft.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.InvalidOperatingState`	Die Instanz ist nicht im Betriebszustand OFF.
	`ERuntimeError-Code.InvalidArchivePath`	Der Archiv-Pfad ist ungültig.
	`ERuntimeError-Code.CreateDirectoriesFailed`	Das Verzeichnis für die ZIP-Datei konnte nicht erstellt werden.
	`ERuntimeError-Code.ArchiveStorageNotCreated`	Die ZIP-Datei konnte nicht erstellt werden.

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- 111 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 Instanz läuft.	
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.

Tabelle 7- 112 RetrieveStorage() - .NET (C#)

Syntax	`void RetrieveStorage( string in_ArchiveStorageFile );`	
Parameter	• `string in_ArchiveStorageFile:` Der vollständige Pfad zur ZIP-Datei.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.InvalidOperatingState`	Die Instanz ist nicht im Betriebszustand OFF.
	`ERuntimeError-Code.InvalidArchivePath`	Der Archiv-Pfad ist ungültig.
	`ERuntimeError-Code.DeleteExistingStorageFailed`	Der alte Speicher kann nicht gelöscht werden.
	`ERuntimeError-Code.RetrieveStorageFailure`	Das ZIP-Datei kann nicht entpackt werden.

7.6.3 Betriebszustand

GetOperatingState() / OperatingState { get; }

Liefert den Betriebszustand des virtuellen Controllers zurück. Wenn sich der Betriebszustand ändert, wird das Ereignis OnOperatingStateChanged() (Seite 239) ausgelöst. Details zum Betriebszustand siehe Datentypen (Seite 301).

Tabelle 7- 113 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_SHUTTING_DOWN:` wenn `PowerOff()` aufgerufen wurde, aber sich der virtuelle Controller noch in der Shutdown-Phase befindet.

Tabelle 7- 114 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.ShuttingDown`: wenn `PowerOff()` aufgerufen wurde, aber sich der virtuelle Controller noch in der Shutdown-Phase befindet.

PowerOn()

Die Funktion erzeugt den Prozess für die Simulation Runtime Instanz und startet die Firmware des virtuellen Controllers.

Tabelle 7- 115 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. Evtl. wird die Länge der `DSTORAGE_PATH_MAX_LENGTH` Zeichen überschritten.
	`SREC_WARNING_ALREADY_EXISTS`	Nur eine Warnung. Die Instanz ist gestartet.
	`SREC_WARNING_TRIAL_MODE_ACTIVE`	Keine Lizenz verfügbar. Sie können die Instanz ohne Einschränkung für die Dauer einer Stunde nutzen. Danach wird die Instanz abgeschaltet. Die Warnung erfolgt auch dann, wenn Sie innerhalb dieser Stunde eine Lizenz einspielen. Die Instanz wird dann nicht abgeschaltet.
	`SREC_VIRTUAL_SWITCH_MISCONFIGURED`	Der virtuelle Switch ist falsch konfiguriert.
	`SREC_INSTANCE_NOT_RUNNING`	Der Prozess des virtuellen Controllers läuft nicht mehr.

Tabelle 7- 116 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.
	`ERuntimeError-Code.WarningAlreadyExists`	Nur eine Warnung. Die Instanz ist gestartet.
	`ERuntimeError-Code.WarningTrialModeActive`	Keine Lizenz verfügbar. Sie können die Instanz ohne Einschränkung für die Dauer einer Stunde nutzen. Danach wird die Instanz abgeschaltet. Die Warnung erfolgt auch dann, wenn Sie innerhalb dieser Stunde eine Lizenz einspielen. Die Instanz wird dann nicht abgeschaltet.
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`ERuntimeError-Code.ErrorLoadingDll`	Die Siemens.Simatic.Simulation. Runtime.Instance.exe kann die Siemens.Simatic.PlcSim.Vplc1500.dll nicht laden.
	`ERuntimeError-Code.StoragePathAlreadyInUse`	Der ausgewählte Pfad für diese Instanz wird bereits von einer anderen Instanz genutzt.
	`ERuntimeError-Code.NoStoragePathSet`	Der Pfad konnte nicht erstellt werden. Evtl. wird die Länge der `DSTORAGE_PATH_MAX_LENGTH` Zeichen überschritten.
	`ERuntimeError-Code.VirtualSwitchMisconfigured`	Der virtuelle Switch ist falsch konfiguriert.
	`ERuntimeError-Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht mehr.

PowerOff()

Fährt die Simulation Runtime herunter und schließt deren Prozess.

Tabelle 7- 117 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. Erwartete Betriebszustände, 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- 118 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. Erwartete Betriebszustände, wenn diese Funktion erfolgreich ist: `{ EOperatingState.Off }`	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`ERuntimeError-` `Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

MemoryReset()

Fährt den virtuellen Controller herunter, schließt dessen Prozesse und führt einen Neustart durch.

Tabelle 7- 119 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- 120 MemoryReset() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2"><pre>void MemoryReset();
void MemoryReset(
 UInt32 in_Timeout_ms
);</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.

Erwartete Betriebszustände, wenn diese Funktion erfolgreich ist:
<code>{ EOperatingState.Run, 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>ERuntimeError-
Code.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>ERuntimeError-
Code.InstanceNotRunning</code></td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
</table>

Run()

Fordert vom virtuellen Controller, in den Betriebszustand RUN zu wechseln.

Tabelle 7- 121 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- 122 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`ERuntimeError-Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

Stop()

Fordert vom virtuellen Controller, in den Betriebszustand STOP zu wechseln.

Tabelle 7- 123 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. Erwartete Betriebszustände, 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- 124 Stop() - .NET (C#)

Syntax	`void Stop();`
	`void Stop(`
	`  bool in_IsSynchronous`
	`);`	

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.Stop }`	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Der erwartete Betriebszustand tritt nicht rechtzeitig ein.
	`ERuntimeError-` `Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

7.6.4 Variablentabelle

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]", wobei {a} und {b} dem ersten Index der jeweiligen Dimension entsprechen

..

Entry_N: "ArrayName[x,y]", wobei {x} und {y} dem letzten Index der jeweiligen Dimension entsprechen

Für die Liste ist Speicher für bis zu 500000 Einträgen (nicht PLC-Variablen) reserviert. 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- 125 UpdateTagList() - Native C++

Syntax	`ERuntimeErrorCode UpdateTagList();`
	`ERuntimeErrorCode UpdateTagList(`
	`  ETagListDetails in_TagListDetails`
	`);`
	`ERuntimeErrorCode UpdateTagList(`
	`  ETagListDetails in_TagListDetails,`
	`  bool in_IsHMIVisibleOnly`
	`);`	

Parameter	• `ETagListDetails in_TagListDetails:` Jede Kombination der folgenden vier Bereiche: **IO**: Eingänge und Ausgänge **M**: Merker **CT**: Zähler und Zeiten **DB**: Datenbausteine **Beispiel: IOM** liest nur die Variablen aus der Area Eingänge / Ausgänge und Merker. Die Voreinstellung ist **IOMCTDB**. • `bool in_IsHMIVisibleOnly:` Wenn `true`, werden nur Variablen gelesen, die mit "HMI Visible" markiert sind. Die Voreinstellung ist `true`.	
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.

Tabelle 7- 126 UpdateTagList() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2">

```
void UpdateTagList();
void UpdateTagList(
 ETagListDetails in_TagListDetails
);
void UpdateTagList(
 ETagListDetails in_TagListDetails,
 bool in_IsHMIVisibleOnly
);
```

</td></tr>
<tr><td>Parameter</td><td colspan="2">

- `ETagListDetails in_TagListDetails:`

  Jede Kombination der folgenden vier Bereiche:

  IO: Eingänge und Ausgänge
  M: Merker
  CT: Zähler und Zeiten
  DB: Datenbausteine

  Beispiel: IOM liest nur die Variablen aus der Area Eingänge / Ausgänge und Merker.
  Die Voreinstellung ist **IOMCTDB**.

- `bool in_IsHMIVisibleOnly:`

  Wenn `true`, werden nur Variablen gelesen, die mit "HMI Visible" markiert sind.
  Die Voreinstellung ist `true`.

</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>`ERuntimeError-Code.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>`ERuntimeError-Code.InstanceNotRunning`</td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td></td><td>`ERuntimeError-Code.NotEnoughMemory`</td><td>Es werden mehr als 500000 Einträge angefordert.</td></tr>
</table>

GetTagListStatus()

Liefert den aktuellen Update-Status des Variablentabellen-Speichers zurück.

"`inout_TagListDetails`" ist `NONE`, wenn die Tabelle aktualisiert werden muss.

Tabelle 7- 127 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- 128 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
	`ERuntimeError-` `Code.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- 129 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- 130 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- 131 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

CreateConfigurationFile()

Schreibt alle Einträge aus der Variablentabelle in eine XML-Datei.

Tabelle 7- 132 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- 133 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
	`ERuntimeError-Code.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 I/O-Zugriff über Adresse - Lesen

InputArea { get; }, MarkerArea { get; }, OutputArea { get; }

Liefert eine Schnittstelle zurück, die Sie nutzen, um die folgenden .NET-Funktionen des Kapitels "I/O-Zugriff über Adresse" aufzurufen.

Tabelle 7- 134 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- 135 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 300).
Rückgabewerte	`UINT32`: Größe der Area in Bytes. Wenn die Funktion erfolgreich war, ist der Wert ungleich 0.

Tabelle 7- 136 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!

Empfehlung: Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 137 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 300). • `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- 138 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>`ERuntimeError-Code.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>`ERuntimeError-Code.InstanceNotRunning`</td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td></td><td>`ERuntimeError-Code.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!

Empfehlung: Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 139 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 300). • `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- 140 ReadByte() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2">

```
Byte InputArea.ReadByte(
 UInt32 in_Offset
);
Byte MarkerArea.ReadByte(
 UInt32 in_Offset
);
Byte OutputArea.ReadByte(
 UInt32 in_Offset
);
```

</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.

</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">`Byte:` Bytewert.</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>`ERuntimeError-Code.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>`ERuntimeError-Code.InstanceNotRunning`</td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td></td><td>`ERuntimeError-Code.IndexOutOfRange`</td><td>Offset ist ungültig.</td></tr>
</table>

ReadBytes()

Liest ein Byte-Array aus der Area.

Hinweis

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Empfehlung: Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 141 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 300). • `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- 142 ReadBytes() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2">
<pre>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
);</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, der von <code>AreaSize</code> zurückgegeben wird.
<code>UInt32 in_BytesToRead:</code>
Die Anzahl der Bytes, die gelesen werden.

</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2"><code>Byte[]:</code> Die gelesenen Bytes.</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>ERuntimeError-
Code.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>ERuntimeError-
Code.InstanceNotRunning</code></td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td></td><td><code>ERuntimeError-
Code.IndexOutOfRange</code></td><td>Der Offset liegt außerhalb der Area-Größe. Kein Byte konnte gelesen werden.</td></tr>
</table>

ReadSignals()

Liest eine Signalliste im Ganzen (Strukturen und Felder). 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!

Empfehlung: Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 143 ReadSignals() - Native C++

Syntax	`ERuntimeErrorCode ReadSignals(`
	` EArea in_Area,`
	` SDataValueByAddress* inout_Signals,`
	` UINT32 in_SignalCount`
	`);`	

Parameter	• `EArea in_Area:` Die Area, von der Sie lesen möchten. Zulässige Werte: `{SRA_INPUT, SRA_MARKER, SRA_OUTPUT}`. Siehe EArea (Seite 300). • `SDataValueByAddress* inout_Signals:` Die Signalliste, die gelesen wird. Das Ergebnis wird in der Struktur gespeichert. • `UINT32 in_SignalCount:` Die Anzahl der Signale in der Liste.	
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- 144 ReadSignals() - .NET (C#)

Syntax	`void ReadSignals(`
	`  ref SDataValueByAddress[] inout_Signals`
	`);`	

Parameter	• `ref SDataValueByAddress[] inout_Signals:` Die Signalliste, die gelesen wird.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeError-` `Code.IndexOutOfRange`	Offset oder Bits sind ungültig.

7.6.5.2 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!

Empfehlung: Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 145 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 300). • `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- 146 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeError-Code.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!

Empfehlung: Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 147 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 300). • `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- 148 WriteByte() - .NET (C#)

Syntax	`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`
	`);`	

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_Value:` Bytewert.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeError-Code.IndexOutOfRange`	Offset ist ungültig.

WriteBytes()

Schreibt ein Byte-Array in die Area.

Hinweis

Daten können überschrieben werden

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Empfehlung: Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 149 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 300). • `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- 150 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>ERuntimeError-
Code.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>ERuntimeError-
Code.InstanceNotRunning</code></td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td></td><td><code>ERuntimeError-
Code.IndexOutOfRange</code></td><td>Der Offset liegt außerhalb der Area-Größe. Kein Byte konnte geschrieben werden.</td></tr>
</table>

WriteSignals()

Schreibt eine Signalliste im Ganzen (Strukturen und Felder). Die Funktion berücksichtigt auch die Byte-Reihenfolge (Endianness).

Die Funktion unterstützt nur primitive Datentyp-Signale. Die Funktion ist aber nicht typsicher.

Hinweis

Daten können überschrieben werden

Die Funktion erlaubt Zugriff auf den gesamten Speicherbereich des virtuellen Controllers!

Empfehlung: Nutzen Sie daher den Zugriff über den Variablennamen und nicht über die Adressbereiche.

Tabelle 7- 151 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 300). • `SDataValueByAddress* inout_Signals:` Die Signalliste, die geschrieben wird. • `UINT32 in_SignalCount:` Die Anzahl der Signale in der Liste.	
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- 152 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	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeError-` `Code.IndexOutOfRange`	Offset oder Bits sind ungültig.

7.6.5.3 I/O-Zugriff über Variablenname - Lesen

Read()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 153 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 mit ReadSignals() im Ganzen 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 154 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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.
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadBool()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 155 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 156 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadInt8()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 157 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 158 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadInt16()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 159 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 160 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadInt32()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 161 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 162 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadInt64()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 163 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 164 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadUInt8()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 165 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 166 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadUInt16()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 167 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 168 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadUInt32()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 169 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 170 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadUInt64()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 171 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 172 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadFloat()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 173 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.
	`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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 174 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadDouble()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 175 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.
	`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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 176 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadChar()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 177 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.
	`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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 178 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadWChar()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 179 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.
	`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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 180 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

ReadSignals()

Liest mehrere Signale innerhalb eines API-Aufrufs.

Reads multiple signals within a single API call.

Tabelle 7- 181 ReadSignals() - Native C++

Syntax	`ERuntimeErrorCode ReadSignals(`
	`  SDataValueByName* inout_Signals,`
	`  UINT32 in_SignalCount`
	`);`	

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. • `UINT32 in_SignalCount:` Die Anzahl der Signale, die 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 182 ReadSignals() - .NET (C#)

Syntax	`void ReadSignals(`
	`  ref SDataValueByName[] inout_Signals`
	`)`	

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.	
Rückgabewerte	`SDataValue`: Enthält den Wert und den Typ der PLC-Variablen.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

7.6.5.4 I/O-Zugriff über Variablenname - Schreiben

Write()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 183 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 mit `WriteSignals()` im Ganzen 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.
	`SREC_WRONG_ARGUMENT`	Der erwartete Typ ist `UNSPECIFIC`.

Tabelle 7- 184 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 mit `WriteSignals()` im Ganzen geschrieben werden.	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.
	`ERuntimeErrorCode.WrongArgument`	Der erwartete Typ ist `UNSPECIFIC`.

WriteBool()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 185 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 186 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.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.
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteInt8()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 187 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 188 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteInt16()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 189 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 190 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteInt32()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 191 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 192 WriteInt32() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2"><pre>void WriteInt32(
 string in_Tag
 Int32 in_Value
)</pre></td></tr>
<tr><td>Parameter</td><td colspan="2">• <code>string in_Tag:</code>
Der Name der PLC-Variablen, die geschrieben werden soll.
• <code>Int32 in_Value:</code>
Enthält den Wert der PLC-Variablen.</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">Keine</td></tr>
<tr><td rowspan="8">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>ERuntimeError-
Code.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>ERuntimeError-
Code.InstanceNotRunning</code></td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td><code>ERuntimeErrorCode.DoesNotExist</code></td><td>Der Eintrag existiert nicht in der gespeicherten Variablentabelle.</td></tr>
<tr><td><code>ERuntimeErrorCode.NotSupported</code></td><td>Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.</td></tr>
<tr><td><code>ERuntimeErrorCode.TypeMissmatch</code></td><td>Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).</td></tr>
<tr><td><code>ERuntimeErrorCode.NotUpToData</code></td><td>Die gespeicherte Variablentabelle muss aktualisiert werden.</td></tr>
</table>

WriteInt64()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 193 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 194 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteUInt8()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 195 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 196 WriteUInt8() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2"><pre>void WriteUInt8(
 string in_Tag
 UInt8 in_Value
)</pre></td></tr>
<tr><td>Parameter</td><td colspan="2">• <code>string in_Tag:</code>
Der Name der PLC-Variablen, die geschrieben werden soll.
• <code>UInt8 in_Value:</code>
Enthält den Wert der PLC-Variablen.</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">Keine</td></tr>
<tr><td rowspan="8">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>ERuntimeError-
Code.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>ERuntimeError-
Code.InstanceNotRunning</code></td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td><code>ERuntimeErrorCode.DoesNotExist</code></td><td>Der Eintrag existiert nicht in der gespeicherten Variablentabelle.</td></tr>
<tr><td><code>ERuntimeErrorCode.NotSupported</code></td><td>Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.</td></tr>
<tr><td><code>ERuntimeErrorCode.TypeMissmatch</code></td><td>Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).</td></tr>
<tr><td><code>ERuntimeErrorCode.NotUpToData</code></td><td>Die gespeicherte Variablentabelle muss aktualisiert werden.</td></tr>
</table>

WriteUInt16()

Liest den Wert einer PLC-Variablen.

Tabelle 7- 197 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 198 WriteUInt16() - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="2"><pre>void WriteUInt16(
 string in_Tag
 UInt16 in_Value
)</pre></td></tr>
<tr><td>Parameter</td><td colspan="2">• <code>string in_Tag:</code>
Der Name der PLC-Variablen, die geschrieben werden soll.
• <code>UInt16 in_Value:</code>
Enthält den Wert der PLC-Variablen.</td></tr>
<tr><td>Rückgabewerte</td><td colspan="2">Keine</td></tr>
<tr><td rowspan="8">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>ERuntimeError-
Code.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>ERuntimeError-
Code.InstanceNotRunning</code></td><td>Der Prozess des virtuellen Controllers läuft nicht.</td></tr>
<tr><td><code>ERuntimeErrorCode.DoesNotExist</code></td><td>Der Eintrag existiert nicht in der gespeicherten Variablentabelle.</td></tr>
<tr><td><code>ERuntimeErrorCode.NotSupported</code></td><td>Der Zugriff auf ganze Strukturen oder Felder wird nicht unterstützt.</td></tr>
<tr><td><code>ERuntimeErrorCode.TypeMissmatch</code></td><td>Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).</td></tr>
<tr><td><code>ERuntimeErrorCode.NotUpToData</code></td><td>Die gespeicherte Variablentabelle muss aktualisiert werden.</td></tr>
</table>

WriteUInt32()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 199 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 200 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteUInt64()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 201 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 202 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteFloat()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 203 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 204 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteDouble()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 205 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 206 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteChar()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 207 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 208 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteWChar()

Schreibt den Wert einer PLC-Variablen.

Tabelle 7- 209 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_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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.

Tabelle 7- 210 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.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.TypeMissmatch`	Der erwartete Typ passt nicht zum gespeicherten Typ. Siehe Kompatible primitive Datentypen (Seite 308).
	`ERuntimeErrorCode.NotUpToData`	Die gespeicherte Variablentabelle muss aktualisiert werden.

WriteSignals()

Schreibt mehrere Signale innerhalb eines API Aufrufs.

Tabelle 7- 211 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.	
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 308).
	`SREC_NOT_UP_TO_DATE`	Die gespeicherte Variablentabelle muss aktualisiert werden.
	`SREC_WRONG_ARGUMENT`	Der erwartete Typ ist `UNSPECIFIC`.

Tabelle 7- 212 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	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.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.
	`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- 213 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- 214 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- 215 SystemTime { get; set; } - .NET (C#)

Syntax	`DateTime SystemTime { get; set; }`	
Parameter	Keine	
Rückgabewerte	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.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- 216 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- 217 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- 218 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
	`ERuntimeError-Code.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 des virtuellen Controllers zurück.

Tabelle 7- 219 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 Zykluskontrollpunkt wirksam.

Tabelle 7- 220 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 Zykluskontrollpunkt wirksam.

Tabelle 7- 221 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
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.

SetAlwaysSendOnEndOfCycleEnabled()

Setzt den Modus `AlwaysSendOnEndOfCycle`. Wenn der Modus gesetzt ist, wird für jede Betriebsart nach jedem Zyklusende das Ereignis `OnEndOfCycle` ausgelöst. Wenn das Ereignis auch in der Betriebsart `Default` empfangen werden soll, setzen Sie den Rückgabewert auf `true`. Siehe OnEndOfCycle (Seite 244).

Tabelle 7- 222 SetAlwaysSendOnEndOfCycleEnabled() - Native C++

Syntax	`ERuntimeErrorCode SetAlwaysSendOnEndOfCycleEnabled( bool in_Enable );`	
Parameter	• `bool in_Enable`: Wenn `true`, dann wird das Ereignis `OnEndOfCycle` nach jedem Zyklus 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.

IsAlwaysSendOnEndOfCycleEnabled()

Liefert den Modus `AlwaysSendOnEndOfCycle` zurück. Wenn die Funktion fehlschlägt, ist der Rückgabewert `false`.

Tabelle 7- 223 IsAlwaysSendOnEndOfCycleEnabled() - Native C++

Syntax	`bool IsAlwaysSendOnEndOfCycleEnabled();`
Parameter	Keine
Rückgabewerte	• `false`: Das Ereignis wird nicht ausgelöst (außer der Modus Sync-Freeze ist aktiv). • `true`: Das Ereignis wird nach jedem Zyklus ausgelöst.

IsAlwaysSendOnEndOfCycleEnabled { get; set; }

Liefert oder setzt den Modus `AlwaysSendOnEndOfCycle`. Wenn der Modus gesetzt ist, wird für jede Betriebsart nach jedem Zyklusende das Ereignis `OnEndOfCycle` ausgelöst. Wenn das Ereignis auch in der Betriebsart `Default` empfangen werden soll, setzen Sie den Rückgabewert auf `true`. Siehe OnEndOfCycle (Seite 244).

Tabelle 7- 224 IsAlwaysSendOnEndOfCycleEnabled { get; set; } - .NET (C#)

Syntax	`bool IsAlwaysSendOnEndOfCycleEnabled { 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
	`ERuntimeError-Code.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 der Betriebsart `ExtendedSingleStep` verwendet wird.

Tabelle 7- 225 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 der Betriebsart `ExtendedSingleStep` 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- 226 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 der Betriebsart `ExtendedSingleStep` 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- 227 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
	`ERuntimeError-Code.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.

RunNextCycle()

Wenn der virtuelle Controller in der Betriebsart `SingleStep` oder `ExtendedSingleStep` betrieben wird, wird er am Zykluskontrollpunkt angehalten (Freeze-Zustand). Mit der Funktion `RunNextCycle()` wird der nächste Zyklus getriggert.

Tabelle 7- 228 RunNextCycle() - Native C++

Syntax	`ERuntimeErrorCode RunNextCycle();`	
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- 229 RunNextCycle() - .NET (C#)

Syntax	`void RunNextCycle();`	
Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.

StartProcessing()

Wenn der virtuelle Controller in der Betriebsart `TimespanSynchronized` läuft, wird er am Zykluskontrollpunkt angehalten (Freeze-Zustand). Die Funktion `StartProcessing()` weckt den virtuellen Controller aus dem Freeze-Zustand. Der virtuelle Controller läuft nun mindestens solange wie angefordert, bevor er beim nächsten Zykluskontrollpunkt in den Freeze-Zustand wechselt (anhält).

Tabelle 7- 230 StartProcessing() - Native C++

Syntax	`ERuntimeErrorCode StartProcessing(`
	`  INT64 in_MinimalTimeToRun_ns`
	`);`	

Parameter	• `INT64 in_CycleTime_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- 231 StartProcessing() - .NET (C#)

Syntax	`void StartProcessing(`
	`  Int64 in_MinimalTimeToRun_ns`
	`);`	

Parameter	• `Int64 in_CycleTime_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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Instanz ist im Runtime Manager nicht registriert.
	`ERuntimeErrorCode.Timeout`	Die Funktion kehrt nicht rechtzeitig wieder.
	`ERuntimeError-` `Code.InstanceNotRunning`	Der Prozess des virtuellen Controllers läuft nicht.
	`ERuntimeErrorCode.WrongArgument`	Der Wert ist kleiner als 0.

Weitere Informationen

Weitere Informationen siehe Kapitel Zeitverhalten (Seite 64), Simulation anhalten (Seite 67)

Siehe auch

OnOperatingStateChanged (Seite 239)

7.6.8 Ereignisse

Ereignisse für IInstances

In der API werden folgende Ereignisse unterschieden:

Tabelle 7- 232 Ereignisse für die Schnittstelle IInstances

Ereignis	Ursache
OnOperatingStateChanged (Seite 239)	Dieses Ereignis wird immer ausgelöst, wenn sich der Betriebszustand des virtuellen Controllers geändert hat.
EndOfCycle (Seite 244)	Dieses Ereignis wird ausgelöst, wenn der virtuelle Controller das Ende des Hauptzyklus erreicht. Wenn der virtuelle Controller in der Betriebsart Default betrieben wird, muss das Flag `AlwaysSendOnEndOfCycle` gesetzt werden, um das Ereignis zu empfangen. Siehe AlwaysSendOnEndOfCycle (Seite 232).
OnConfigurationChanging (Seite 247)	Dieses Ereignis wird ausgelöst, wenn sich die Konfiguration des virtuellen Controllers zu ändern beginnt: • Beim Hochlauf aus der Virtual SIMATIC Memory Card • Zu Beginn eines Downloads Wenn dieses Ereignis ausgelöst wird, wird die gespeicherte Variablentabelle zurückgesetzt.
OnConfigurationChanged (Seite 250)	Dieses Ereignis wird ausgelöst, wenn sich die Konfiguration des virtuellen Controllers geändert hat: • Nach dem Hochlauf aus der Virtual SIMATIC Memory Card • Am Ende eines Downloads • Wenn sich die IP-Adresse ändert
OnLedChanged (Seite 253)	Dieses Ereignis wird ausgelöst, wenn sich die LED-Anzeige des virtuellen Controllers geändert hat.

7.6.8.1 OnOperatingStateChanged

OnOperatingStateChanged

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 233 OnOperatingStateChanged - .NET (C#)

Syntax	`event Delegate_II_EREC_DT_EOS_EOS OnOperatingStateChanged;`
Parameter	Keine. Siehe Delegate_II_EREC_DT_EOS_EOS (Seite 281).
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- 234 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 275).
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- 235 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->WaitForOnOperatingStateChangedEvent(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->WaitForOnOperatingStateChangedEvent();`
	
	`// Do Something`
	`…`
	`}`

UnregisterOnOperatingStateChangedCallback()

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

Tabelle 7- 236 UnregisterOnOperatingStateChangedCallback() - Native C++

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

UnregisterOnOperatingStateChangedEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 237 UnregisterOnOperatingStateChangedEvent() - Native C++

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

Tabelle 7- 238 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- 239 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- 240 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.

7.6.8.2 OnEndOfCycle

OnEndOfCycle

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 241 OnEndOfCycle - .NET (C#)

Syntax	`event Delegate_II_EREC_DT OnEndOfCycle;`
Parameter	Keine. Siehe Delegate_II_EREC_DT (Seite 280).
Rückgabewerte	Keine
Ausnahmen	Keine
Hinweis	Die Event-Handler Methode läuft in einem separaten Thread.

RegisterOnEndOfCycleCallback()

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- 242 RegisterOnEndOfCycleCallback() - Native C++

Syntax	`void RegisterOnEndOfCycleCallback(`
	`  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 274).
Rückgabewerte	Keine
Hinweis	Die Callback-Funktion läuft in einem separaten Thread.

RegisterOnEndOfCycleEvent()

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- 243 RegisterOnEndOfCycleEvent() - Native C++

Syntax	`void RegisterOnEndOfCycleEvent();`
	`void RegisterOnEndOfCycleEvent(`
	`  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

UnregisterOnEndOfCycleCallback()

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

Tabelle 7- 244 UnregisterOnEndOfCycleCallback() - Native C++

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

UnregisterOnEndOfCycleEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 245 UnregisterOnEndOfCycleEvent() - Native C++

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

Tabelle 7- 246 UnregisterOnEndOfCycleEvent() - .NET (C#)

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

WaitForOnEndOfCycleEvent()

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

Tabelle 7- 247 WaitForOnEndOfCycleEvent() - Native C++

Syntax	`bool WaitForOnEndOfCycleEvent();`
	`bool WaitForOnEndOfCycleEvent(`
	`  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- 248 WaitForOnEndOfCycleEvent() - .NET (C#)

Syntax	`bool WaitForOnEndOfCycleEvent();`
	`bool WaitForOnEndOfCycleEvent(`
	`  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.

7.6.8.3 OnConfigurationChanging

OnConfigurationChanging

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 249 OnConfigurationChanging - .NET (C#)

Syntax	`event Delegate_II_EREC_DT OnConfigurationChanging;`
Parameter	Keine. Siehe Delegate_II_EREC_DT (Seite 280).
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- 250 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 274).
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- 251 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- 252 UnregisterOnConfigurationChangingCallback() - Native C++

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

UnregisterOnConfigurationChangingEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 253 UnregisterOnConfigurationChangingEvent() - Native C++

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

Tabelle 7- 254 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- 255 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- 256 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.

7.6.8.4 OnConfigurationChanged

OnConfigurationChanged

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 257 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 285).
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- 258 RegisterOnConfigurationChangedCallback() - Native C++

Syntax	`void RegisterOnConfigurationChangedCallback( EventCallback_SRCC_UINT32_UINT32_INT32 in_CallbackFunction );`
Parameter	• `EventCallback_SRCC_UINT32_UINT32_INT32 in_CallbackFunction:` Eine Callback-Funktion, die ein Ereignis abonniert. Siehe EventCallback_SRCC_UINT32_UINT32_INT32 (Seite 278).
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- 259 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- 260 UnregisterOnConfigurationChangedCallback() - Native C++

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

UnregisterOnConfigurationChangedEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 261 UnregisterOnConfigurationChangedEvent() - Native C++

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

Tabelle 7- 262 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- 263 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- 264 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.

7.6.8.5 OnLedChanged

OnLedChanged

Meldet eine Event-Handler Methode an oder ab.

Tabelle 7- 265 OnLedChanged - .NET (C#)

Syntax	`event Delegate_II_EREC_DT_ELT_ELM OnLedChanged;`
Parameter	Keine. Siehe Delegate_II_EREC_DT_ELT_ELM (Seite 282).
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- 266 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 276).
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- 267 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- 268 UnregisterOnLedChangedCallback() - Native C++

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

UnregisterOnLedChangedEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 269 UnregisterOnLedChangedEvent() - Native C++

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

Tabelle 7- 270 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- 271 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- 272 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.

7.7 API IRemoteRuntimeManager

7.7.1 Schnittstellen - Information und Einstellungen

Dispose()

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

Tabelle 7- 273 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- 274 GetVersion() - Native C++

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

Tabelle 7- 275 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- 276 GetIP() - Native C++

Syntax	`UIP GetIP();`
Parameter	Keine
Rückgabewerte	`UIP`: Liefert die IP-Adresse des PC, auf dem der Runtime Manager läuft.

Tabelle 7- 277 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- 278 GetPort() - Native C++

Syntax	`UINT16 GetPort();`
Parameter	Keine
Rückgabewerte	`UINT16`: Offener Port des PC, auf dem der Remote Runtime Manager läuft.

Tabelle 7- 279 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- 280 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_OT_OF_RANGE`	Das Feld ist zu klein, um den Computernamen zu empfangen.

Tabelle 7- 281 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
	`ERuntimeError-` `Code.InterfaceRemoved`	Die Schnittstelle ist vom Remote Runtime Manager getrennt.
	`ERuntimeError-` `Code.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- 282 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- 283 Disconnect() - .NET (C#)

Syntax	`void Disconnect();`	
Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.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- 284 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- 285 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 292).	
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- 286 RegisterInstanceInfo { get; } - .NET (C#)

Syntax	`SInstanceInfo[] RegisteredInstanceInfo { get; }`	
Parameter	Keine	
Rückgabewerte	Keine	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.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- 287 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 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 286). • `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 88).

Tabelle 7- 288 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 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 286).
Rückgabewerte	Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers. Ansonsten einen Null-Zeiger.

Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.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- 289 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 286). • `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 88).

Tabelle 7- 290 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 286).

Rückgabewerte	Wenn die Funktion erfolgreich ist, eine Schnittstelle eines virtuellen Controllers, ansonsten ein Null-Zeiger.	
Ausnahmen	`Siemens.Simatic.Simulation.Runtime.SimulationRuntimeException`	
	Runtime Fehlercode	Bedingung
	`ERuntimeError-Code.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- 291 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 88)

Tabelle 7- 292 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
	`ERuntimeError-Code.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

7.7.3.1 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- 293 OnConnectionLost - .NET (C#)

Syntax	`event Delegate_IRRTM OnConnectionLost;`
Parameter	Keine. Siehe Delegate_IRRTM (Seite 283)
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- 294 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 277).
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- 295 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- 296 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- 297 UnregisterOnConnectionLostCallback() - Native C++

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

UnregisterOnConnectionLostEvent()

Meldet das Event-Objekt ab.

Tabelle 7- 298 UnregisterOnConnectionLostEvent() - Native C++

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

Tabelle 7- 299 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- 300 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- 301 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 S7-PLCSIM Advanced V1.0 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 von STEP 7 V14 (https://support.industry.siemens.com/cs/document/109011420/step-7-professional-v13-1?dti=0&pnid=14673&lc=en-WW // XmlEditor.InternalXmlClipboard:52c35660-6164-fbfd-288e-e4368e2843f5).

7.8.1 DLL-Importfunktionen (Native C++)

7.8.1.1 ApiEntry_Initialize

Beschreibung

Typ des zentralen Eintrittspunkts für die API DLL.

Tabelle 7- 302 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 initialsiert 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 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 88).

Tabelle 7- 303 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- 304 EventCallback_VOID - Native C++

Syntax	`typedef void (*EventCallback_VOID)();`
Parameter	Keine
Rückgabewerte	Keine

7.8.2.2 EventCallback_II_SREC_ST

Beschreibung

Tabelle 7- 305 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

7.8.2.3 EventCallback_II_SREC_ST_SROS_SROS

Beschreibung

Tabelle 7- 306 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	Runntime Fehlercode	Bedingung
	`SREC_OK`	Die Funktion ist erfolgreich.
	`SREC_WARNING_TRIAL_MODE_ACTIVE`	Keine Lizenz verfügbar. Sie können die Instanz ohne Einschränkung für die Dauer einer Stunde nutzen. Danach wird die Instanz abgeschaltet.
	`SREC_LICENSE_NOT_FOUND`	Der Testmodus ist abgelaufen.
	`SREC_COMMUNICATION_INTERFACE_NOT_AVAILABLE`	Ein Problem mit der ausgewählten Kommunikations-Schnittstelle ist aufgetreten. Überprüfen Sie Ihre Einstellungen.

7.8.2.4 EventCallback_II_SREC_ST_SRLT_SRLM

Beschreibung

Tabelle 7- 307 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

7.8.2.5 EventCallback_II_SREC_ST_INT64_UINT32

Beschreibung

Tabelle 7- 308 EventCallback_II_SREC_ST_INT64_UINT32 - Native C++

Syntax	`typedef void (*EventCallback_II_SREC_ST_INT64_UINT32)(`
	` IInstance* in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` SYSTEMTIME in_SystemTime,`
	` INT64 in_CycleTime_ns,`
	` UINT32 in_CycleCount,`
	`);`

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. • `INT64 in_CycleTime_ns:` Die virtuelle Zeit (in Nanosekunden) seit dem letzten Zykluskontrollpunkt. Wenn die Ereignisse schneller ausgelöst als sie empfangen werden, dann werden mehrere Ereignisse zu einem zusammengefasst. In diesem Fall enthält dieser Wert die virtuelle Zeit (in Nanosekunden), seit das letzte Ereignis empfangen wurde. • `UINT32 in_CycleCount:` Die Anzahl der Zyklen seit dem letzten Zykluskontrollpunkt. 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

7.8.2.6 EventCallback_IRRTM

Beschreibung

Tabelle 7- 309 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

7.8.2.7 EventCallback_SRCC_UINT32_UINT32_INT32

Beschreibung

Tabelle 7- 310 EventCallback_SRCC_UINT32_UINT32_INT32 - Native C++

Syntax	`ERuntimeConfigChanged in_RuntimeConfigChanged,`
	`  UINT32 in_Param1,`
	`  UINT32 in_Param2,`
	`  INT32 in_Param3`
	`);`			

Parameter	`ERuntimeCon-figChanged 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			

7.8.2.8 EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32

Beschreibung

Tabelle 7- 311 EventCallback_II_SREC_ST_SRICC_UINT32_UINT32_UINT32_UINT32 - Native C++

<table>
<tr><td>Syntax</td><td>

```
typedef void
(*EventCallback_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
);
```

</td></tr>
<tr><td>Parameter</td><td>

<code>IInstance in_Sender:</code>
Eine Schnittstelle der Instanz, die dieses Ereignis empfängt.
<code>ERuntimeErrorCode in_ErrorCode:</code>
Ein möglicher Fehlercode.
<code>SYSTEMTIME in_SystemTime:</code>
Die virtuelle Systemzeit des virtuellen Controllers, zum Zeitpunkt als dieses Ereignis ausgelöst wurde.

<table>
<tr><td><code>EInstanceConfigChanged in_InstanceConfigChanged</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><code>SRICC_HARDWARE_SOFTWARE_CHANGED</code></td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td><code>SRICC_IP_CHANGED</code></td><td>Die ID der Schnittstelle</td><td>Die neue IP</td><td>Die neue Subnetzmaske</td><td>Das neue Standard-Gateway</td></tr>
</table>
</td></tr>
<tr><td>Rückgabewerte</td><td>Keine</td></tr>
</table>

7.8.3 Delegat Definitionen (Managed Code)

7.8.3.1 Delegate_Void

Beschreibung

Tabelle 7- 312 Delegate_Void - .NET (C#)

Syntax	`delegate void Delegate_Void();`
Parameter	Keine
Rückgabewerte	Keine

7.8.3.2 Delegate_II_EREC_DT

Beschreibung

Tabelle 7- 313 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

7.8.3.3 Delegate_II_EREC_DT_EOS_EOS

Beschreibung

Tabelle 7- 314 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	Runntime Fehlercode	Bedingung
	`ERuntimeErrorCode.OK`	Die Funktion ist erfolgreich.
	`ERuntimeError-Code.WarningTrialModeActive`	Keine Lizenz verfügbar. Sie können die Instanz ohne Einschränkung für die Dauer einer Stunde nutzen. Danach wird die Instanz abgeschaltet.
	`ERuntimeError-Code.LicenseNotFound`	Der Testmodus ist abgelaufen.
	`ERuntimeError-Code.CommunicationInterfaceNotAvailable`	Ein Problem mit der ausgewählten Kommunikations-Schnittstelle ist aufgetreten. Überprüfen Sie Ihre Einstellungen.

7.8.3.4 Delegate_II_EREC_DT_ELT_ELM

Beschreibung

Tabelle 7- 315 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

7.8.3.5 Delegate_II_EREC_DT_INT64_UINT32

Beschreibung

Tabelle 7- 316 Delegate_II_EREC_DT_INT64_UINT32 - .NET (C#)

Syntax	`delegate void Delegate_II_EREC_DT_INT64_UINT32(`
	` IInstance in_Sender,`
	` ERuntimeErrorCode in_ErrorCode,`
	` DateTime in_DateTime,`
	` Int64 in_CycleTime_ns,`
	` UInt32 in_CycleCount,`
	`);`

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. • `Int64 in_CycleTime_ns:` Die virtuelle Zeit (in Nanosekunden) seit dem letzten Zykluskontrollpunkt. Wenn die Ereignisse schneller ausgelöst als sie empfangen werden, dann werden mehrere Ereignisse zu einem zusammengefasst. In diesem Fall enthält dieser Wert die virtuelle Zeit (in Nanosekunden), seit das letzte Ereignis empfangen wurde. • `UInt32 in_CycleCount:` Die Anzahl der Zyklen seit dem letzten Zykluskontrollpunkt. 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

7.8.3.6 Delegate_IRRTM

Beschreibung

Tabelle 7- 317 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

7.8.3.7 Delegate_SRCC_UINT32_UINT32_INT32

Beschreibung

Tabelle 7- 318 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			

7.8.3.8 Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32

Beschreibung

Tabelle 7- 319 Delegate_II_EREC_DT_SRICC_UINT32_UINT32_UINT32_UINT32 - .NET (C#)

<table>
<tr><td>Syntax</td><td colspan="5">

```
delegate void Delega-
te_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
);
```

</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.

<table>
<tr><td><code>EIn-
stanceCon-
figChanged
in_Instance
ConfigChan-
ged</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><code>Hard-
wareSoft-
wareChanged</code></td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td><code>IPChanged</code></td><td>Die ID der Schnittstelle</td><td>Die neue IP</td><td>Die neue Sub-netzmaske</td><td>Das neue Standard-Gateway</td></tr>
</table>
</td></tr>
<tr><td>Rückgabewerte</td><td colspan="5">Keine</td></tr>
</table>

7.8.4 Definitionen und Konstanten

Folgende Bezeichner werden in der API verwendet:

Tabelle 7- 320 Definitionen und Konstanten - 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.
DCONTROLLER_SHORT_DESIGNATION_MAX_LENGTH	32	Die maximale Länge der Kurzbezeichnung des Controllers (CPU-Typ). Einschließlich der NULL-Terminierung.

Tabelle 7- 321 Definitionen und 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 multi-dimensionalen 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.

7.8.5 Unions (Native C++)

7.8.5.1 UIP

Beschreibung

Enthält eine IPv4-Adresse.

Tabelle 7- 322 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- 323 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

7.8.6.1 SDataValue

Beschreibung

Die Struktur enthält den Wert und den Typ einer PLC-Variablen.

Tabelle 7- 324 SDataValue - Native C++

Syntax	`struct SDataValue`
	`{`
	`UDataValue Value;`
	`EDataType Type;`
	`};`

Member	• `UDataValue Value:` Der Wert der PLC-Variablen • `EPrimitiveDataType Type:` Typ der PLC-Variablen

Tabelle 7- 325 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 SDataValueByAddress

Beschreibung

Diese Struktur repräsentiert eine PLC-Variable, die über ihre Adresse aufgerufen werden kann.

Tabelle 7- 326 SDataValueByAddress - Native C++

Syntax	`struct SDataValueByAddress`
	`{`
	` UINT32 Offset;`
	` UINT8 Bit;`
	` SDataValue DataValue;`
	`};`

Tabelle 7- 327 SDataValueByAddress - .NET (C#)

Syntax	`struct SDataValueByAddress`
	`{`
	` UInt32 Offset;`
	` UInt8 Bit;`
	` SDataValue DataValue;`
	`}`

7.8.6.3 SDataValueByName

Beschreibung

Diese Struktur repräsentiert eine PLC-Variable, die über ihren Namen aufgerufen werden kann.

Tabelle 7- 328 SDataValueByName - Native C++

Syntax	`struct SDataValueByName`
	`{`
	` WCHAR Name[DTAG_NAME_MAX_LENGTH];`
	` SDataValue DataValue;`
	`};`

Tabelle 7- 329 SDataValueByName - .NET (C#)

Syntax	`struct SDataValueByName`
	`{`
	` String Name;`
	` SDataValue DataValue;`
	`}`

7.8.6.4 SConnectionInfo

Beschreibung

Diese Struktur enthält die IP-Adresse und den Port einer TCP/IP-Verbindung.

Tabelle 7- 330 SConnectionInfo - Native C++

Syntax	`struct SConnectionInfo`
	`{`
	` UIP IP;`
	` WORD Port;`
	`};`

Tabelle 7- 331 SConnectionInfo - .NET (C#)

Syntax	`struct SConnectionInfo`
	`{`
	` SIP IP;`
	` UInt16 Port;`
	`}`

7.8.6.5 SInstanceInfo

Beschreibung

Diese Struktur enthält eine IPv4-Adresse.

Tabelle 7- 332 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- 333 SInstanceInfo - .NET (C#)

Syntax	`struct SInstanceInfo`
	`{`
	`  Int32 ID;`
	`  String Name;`
	`}`

Member	• `Int32 ID:` Die ID der Instanz • `String Name:` Der Name der Instanz

7.8.6.6 SDimension

Beschreibung

Diese Struktur enthält Informationen zur Dimension eines Feldes.

Tabelle 7- 334 SDimension - Native C++

Syntax	`struct SDimension`
	`{`
	`  INT32 StartIndex;`
	`  UINT32 Count;`
	`};`

Tabelle 7- 335 SDimension - .NET (C#)

Syntax	`struct SDimension`
	`{`
	`  Int32 StartIndex;`
	`  UInt32 Count;`
	`}`

7.8.6.7 STagInfo

Beschreibung

Diese Struktur enthält Informationen zu einer PLC-Variablen.

Tabelle 7- 336 STagInfo - Native C++

Syntax	`struct STagInfo`
	`{`
	`WCHAR Name[DTAG_NAME_MAX_LENGTH];`
	`EArea Area;`
	`EDataType DataType;`
	`PrimitiveDataType 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 Variable 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- 337 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.

7.8.6.8 SIP

Beschreibung

Diese Struktur enthält eine IPv4-Adresse.

Tabelle 7- 338 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"

7.8.6.9 SIPSuite4

Beschreibung

Diese Struktur enthält eine IPv4-Suite.

Tabelle 7- 339 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- 340 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

7.8.7 Aufzählungen

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`. Fehler werden mit negativen Werten zurückgegeben, Alarme mit positiven.

Tabelle 7- 341 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
};
```

Tabelle 7- 342 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,`
	` MesssageIncomplete = -36,`
	` ArchiveStorageNotCreated = -37,`
	` RetrieveStorageFailure = -38,`
	` InvalidOperatingState = -39,`
	` InvalidArchivePath = -40,`
	` DeleteExistingStorageFailed = -41,`
	` CreateDirectoriesFailed = -42,`
	` NotEnoughMemory = -43,`
	` WarningTrialModeActive = 44,`
	` NotRunning = -45,`
	` NotEmpty = -46,`
	` NotUpToData = -47,`
	` CommunicationInterfaceNotAvailable = -48,`
	` WarningNotComplete = 49,`
	` VirtualSwitchMisconfigured = -50`
	`}`

7.8.7.2 EArea

Beschreibung

Diese Aufzählung enthält alle PLC-Areas, die verfügbare PLC-Variablen beinhalten.

Tabelle 7- 343 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- 344 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- 345 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_ENUMERATION_SIZE = 8`
	`};`

Tabelle 7- 346 EOperatingState - .NET (C#)

Syntax	`enum EOperatingState`
	`{`
	` InvalidOperatingState = 0,`
	` Off = 1,`
	` Booting = 2,`
	` Stop = 3,`
	` Startup = 4,`
	` Run = 5,`
	` Freeze = 6,`
	` ShuttingDown = 7`
	`}`

7.8.7.4 EOperatingMode

Beschreibung

Diese Aufzählung enthält alle Betriebsarten eines virtuellen Controllers.

Tabelle 7- 347 EOperatingMode - Native C++

Syntax	`enum EOperatingMode`
	`{`
	` SROM_DEFAULT = 0,`
	` SROM_SINGLE_STEP = 1,`
	` SROM_EXTENDED_SINGLE_STEP = 2,`
	` SROM_TIMESPAN_SYNCHNRONIZED = 3`
	`};`

Tabelle 7- 348 EOperatingMode - .NET (C#)

Syntax	`enum EOperatingMode`
	`{`
	` Default = 0,`
	` SingleStep = 1,`
	` ExtendedSingleStep = 2,`
	` TimespanSynchronized = 3`
	`}`

7.8.7.5 ECPUType

Beschreibung

Diese Aufzählung enthält alle CPU-Typen, die auf einen virtuellen Controller geladen werden können.

Tabelle 7- 349 ECPUType - Native C++

Syntax	enum ECPUType
	{
	SRCT_1500_Unspecified = 0x000005DC,
	SRCT_1511 = 0x000005E7,
	SRCT_1511v2 = 0x010005E7,
	SRCT_1513 = 0x000005E9,
	SRCT_1513v2 = 0x010005E9,
	SRCT_1515 = 0x000005EB,
	SRCT_1515v2 = 0x010005EB,
	SRCT_1516 = 0x000005EC,
	SRCT_1516v2 = 0x010005EC,
	SRCT_1517 = 0x000005ED,
	SRCT_1518 = 0x000005EE,
	SRCT_1511C = 0x000405E7,
	SRCT_1512C = 0x000405E8,
	SRCT_1511F = 0x000105E7,
	SRCT_1511Fv2 = 0x010105E7,
	SRCT_1513F = 0x000105E9,
	SRCT_1513Fv2 = 0x010105E9,
	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_1517T = 0x000805ED,
	SRCT_1517TF = 0x000905ED,
	SRCT_1518ODK = 0x001005EE,
	SRCT_1518FODK = 0x001105EE,
	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- 350 ECPUType - .NET (C#)

Syntax	enum ECPUType
	{
	CPU1500_Unspecified = 0x000005DC,
	CPU1511 = 0x000005E7,
	CPU1511v2 = 0x010005E7,
	CPU1513 = 0x000005E9,
	CPU1513v2 = 0x010005E9,
	CPU1515 = 0x000005EB,
	CPU1515v2 = 0x010005EB,
	CPU1516 = 0x000005EC,
	CPU1516v2 = 0x010005EC,
	CPU1517 = 0x000005ED,
	CPU1518 = 0x000005EE,
	CPU1511C = 0x000405E7,
	CPU1512C = 0x000405E8,
	CPU1511F = 0x000105E7,
	CPU1511Fv2 = 0x010105E7,
	CPU1513F = 0x000105E9,
	CPU1513Fv2 = 0x010105E9,
	CPU1515F = 0x000105EB,
	CPU1515Fv2 = 0x010105EB,
	CPU1516F = 0x000105EC,
	CPU1516Fv2 = 0x010105EC,
	CPU1517F = 0x000105ED,
	CPU1518F = 0x000105EE,
	CPU1511T = 0x000805E7,
	CPU1515T = 0x000805EB,
	CPU1517T = 0x000805ED,
	CPU1517TF = 0x000905ED,
	CPU1518ODK = 0x001005EE,
	CPU1518FODK = 0x001105EE,
	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 Kommunikations-Schnittstellen eines virtuellen Controllers.

Tabelle 7- 351 ECommunicationInterface - Native C++

Syntax	`enum ECommunicationInterface`
	`{`
	` SRCI_NONE = 0,`
	` SRCI_SOFTBUS = 1,`
	` SRCI_TCPIP = 2,`
	` SRCI_ENUMERATION_SIZE = 3`
	`};`

Tabelle 7- 352 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- 353 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- 354 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- 355 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- 356 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- 357 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- 358 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- 359 Kompatible primitive Datentypen - Lesen

API	PLCSIM Advanced Instanz												
	Bool	INT8	INT16	INT32	INT64	UINT8	UINT16	UINT32	UINT64	Float	Double	Char	WChar
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- 360 Kompatible primitive Datentypen - Schreiben

API	PLCSIM Advanced Instanz												
	Bool	INT8	INT16	INT32	INT64	UINT8	UINT16	UINT32	UINT64	Float	Double	Char	WChar
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				
UINT32								X	X				
UINT64									X				
Float										X			
Double											X		
Char												X	
WChar													X

7.8.7.10 EDataType

Beschreibung

Diese Aufzählung enthält alle CPU-Datentypen (STEP 7).

Tabelle 7- 361 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- 362 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- 363 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- 364 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- 365 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- 366 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- 367 EInstanceConfigChanged - Native C++

Syntax	`enum EInstanceConfigChanged`
	`{`
	` SRICC_HARDWARE_SOFTWARE_CHANGED = 0,`
	` SRICC_IP_CHANGED = 1`
	`};`

Tabelle 7- 368 EInstanceConfigChanged - .NET (C#)

Syntax	`enum EInstanceConfigChanged`
	`{`
	` HardwareSoftwareChanged = 0,`
	` IPChanged = 1`
	`}`

Einschränkungen

8

8.1 Übersicht

Bestimmte Aktionen oder Ereignisse können in S7-PLCSIM Advanced oder in STEP 7 zu einem Verhalten führen, das von dem einer Hardware-CPU abweicht. Einschränkungen und mögliche Abhilfen finden Sie in folgenden Kapiteln:

- OPC UA Server (Seite 316)
- Webserver (Seite 318)
- Einschränkungen bei Kommunikationsdiensten (Seite 319)
- Einschränkungen bei Anweisungen (Seite 320)
- Einschränkungen bei Motion Control (Seite 321)
- Einschränkungen bei lokaler Kommunikation über Softbus (Seite 322)
- Fehler bei Überlauf zyklischer Ereignisse (Seite 324)
- Abweichende E/A-Werte im STEP 7-Anwenderprogramm (Seite 324)
- Mehrfache Simulationen und mögliche Kollision der IP-Adressen (Seite 324)
- Simulation im Standby-Modus (Seite 325)
- Fehler beim Installieren mit Virenscanner von Kaspersky (Seite 325)

8.2 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 Kommunikations-Schnittstelle "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. Die Daten werden aber von PLCSIM Advanced nicht weiterverarbeitet. So ist gewährleistet, dass der Anwender sein Projekt nicht ändern muss, um eine Simulation durchzuführen.

Zertifikate

- **Server-Zertifikat**

  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.

8.3 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.

Die Simulation des Webservers ist unter S7-PLCSIM Advanced V1.0 eingeschränkt.

Jede PLCSIM Advanced Instanz kann ihren eigenen Webserver simulieren.

Der Freeze-Zustand eines virtuellen Controllers wird als interner Betriebszustand nicht angezeigt.

Webserver konfigurieren

S7-PLCSIM Advanced

Starten Sie die Instanzen über die Kommunikations-Schnittstelle "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.

Einschränkte 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.

8.4 Einschränkungen bei Kommunikationsdiensten

TUSEND / TURCV

Wenn Sie die UDP-Bausteine TUSEND und TURCV über die Kommunikations-Schnittstelle "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 Kommunikations-Schnittstelle "PLCSIM Virtual Ethernet Adapter" (TCP/IP) ein.

8.5 Einschränkungen bei Anweisungen

S7-PLCSIM Advanced simuliert Anweisungen für die CPUs S7-1500 und ET 200SP bis auf wenige Ausnahmen, z. B. Alarme.

Manche Anweisungen werden teilweise unterstützt. Bei diesen prüft S7-PLCSIM Advanced 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 von S7-PLCSIM Advanced als nicht betriebsbereit behandelt, ihr Wert ist immer "OK".

S7-PLCSIM Advanced unterstützt die folgenden Anweisungen nicht:

Tabelle 8- 1 Nicht unterstützte Anweisungen

Anweisung	Beschreibung
DIS_AIRT	Alarme mit höherer Priorität und asynchrone Fehler verzögern
EN_AIRT	Alarme mit höherer Priorität und asynchrone Fehler aktivieren
DIS_IRT	Aufruf neuer Alarme und asynchroner Fehler sperren
RE_TRIGR	Zeitüberwachung neu starten
WR_DPARM	Datensatz übertragen
GETIO	Prozessabbild lesen
SETIO	Prozessabbild übertragen
GETIO_PART	Einen Teil der Ausgänge eines DP-Normslaves / PROFINET IO-Devices lesen
SETIO_PART	Einen Teil der Ausgänge eines DP-Normslaves / PROFINET IO-Devices schreiben
DPRD_DAT	Konsistente Daten eines Standard-DP-Slave lesen
DPWR_DAT	Konsistente Daten in einen Standard-DP-Slave schreiben
RD_OBINF	OB Startinformation lesen
DP_TOPOL	DP-Bustopologie
PORT_CFG	Sicherheitsfunktion
ATTACH	Einem Prozessalarm einen OB zuordnen
DETACH	Zuordnung zwischen Prozessalarm und OB lösen

8.6 Einschränkungen bei Motion Control

8.6.1 Motion Control-Ressourcen

Auf jeder CPU gibt es Motion Control-Ressourcen, die auf die Technologieobjekte verteilt werden können.

Hinweis

Maximal 5120 Motion Control-Ressourcen

PLCSIM Advanced unterstützt maximal 5120 Motion Control-Ressourcen. Die Simulation der Bewegungssteuerung ist daher für folgende CPUs eingeschränkt:

Tabelle 8- 2 CPUs mit eingeschränkten Motion Control-Ressourcen

Typ		Artikelnummer
Standard-CPUs	CPU 1517-3 PN/DP	6ES7517-3AP00-0AB0
	CPU 1518-4 PN/DP	6ES7518-4AP00-0AB0
	CPU 1518-4 PN/DP ODK[1]	6ES7518-4AP00-3AB0
Fehlersichere CPUs	CPU 1517F-3 PN/DP	6ES7517-3FP00-0AB0
	CPU 1518F-4 PN/DP	6ES7518-4FP00-0AB0
	CPU 1518F-4 PN/DP ODK[1]	6ES7518-4FP00-3AB0
Technologie-CPUs	CPU 1517T-3 PN/DP[2]	6ES7517-3TP00-0AB0
	CPU 1517TF-3 PN/DP[2]	6ES7517-3UP00-0AB0

[1] Die ODK-Funktionalität dieser CPU wird nicht simuliert.

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

Überschreiten des Mengengerüsts

STEP 7 prüft, ob das Mengengerüst für eine projektierte CPU eingehalten wird, und meldet, wenn es überschritten wird. Wenn Sie ein solches Projekt in eine CPU laden, erfolgt erneut eine Meldung.

Hinweis

Ein Projekt, das mehr als 5120 Motion Control-Ressourcen verwendet, kann auf einen virtuellen Controller heruntergeladen werden. Es erfolgt aber keine Meldung, dass das Mengengerüst überschritten wurde.

Erst wenn das Anwenderprogramm auf die Technologieobjekte zugreift, erkennen Sie an der Fehlermeldung am Baustein und am Wert 0 des Objekts, dass Motion Control Objekte aufgrund einer Überschreitung des Mengengerüsts nicht genutzt werden können.

8.6.2 Technologiemodule

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

Der Download eines 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.

Weitere Informationen

Weitere Informationen zu Motion Control erhalten Sie in folgenden Handbüchern:

- Gerätehandbücher zu den unterstützen SIMATIC Controllern (http://w3.siemens.com/mcms/industrial-automation-systems-simatic/de/handbuchuebersicht/Seiten/Default.aspx)
- Funktionshandbuch S7-1500 Motion Control (https://support.industry.siemens.com/cs/ww/de/view/109739589)
- Funktionshandbuch S7-1500T Motion Control (https://support.industry.siemens.com/cs/ww/de/view/109481326)

8.7 Einschränkungen bei lokaler Kommunikation über Softbus

Identische IP-Adressen für Instanzen

Wenn die Kommunikations-Schnittstelle "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 128)

Online und Diagnose

Wenn die Kommunikations-Schnittstelle "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 125)

8.8 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.

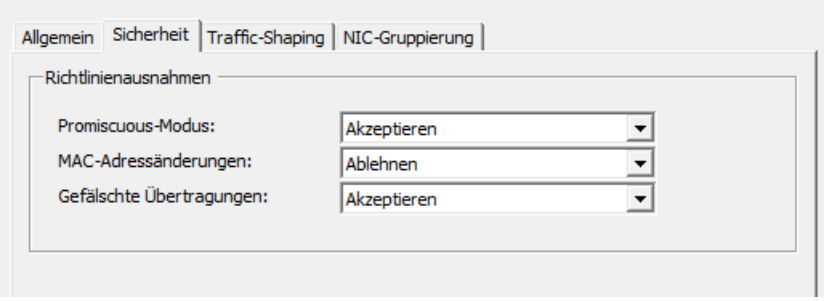


Bild 8-1 Richtlinienausnahmen für VMware vSphere Hypervisor (ESXi)

8.9 Fehler bei Überlauf 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 65)

8.10 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 59)

8.11 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.

8.12 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.

8.13 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 S7-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.

8.14 Fehler beim Installieren mit Virenscanner von Kaspersky

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

Abhilfe

Kontrollieren Sie Ihre Netzwerkeinstellungen wie in Kapitel Verteilte Kommunikation aktivieren (Seite 47) beschrieben.

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)
ES	Engineering System
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.
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
TIA	Totally Integrated Automation
UTC	Coordinated Universal Time
VM	Virtual Machine
VPLC	Virtual Programmable Logic Controller
WinCC	Windows Control Center