

Industry Online Support

NEWS

Programmierung eines OPC DA .NET Clients mit C# für den SIMATIC NET OPC Server (COM/DCOM)

SIMATIC NET OPC Server

https://support.industry.siemens.com/cs/ww/de/view/21043779

Siemens Industry Online Support



Rechtliche Hinweise

Nutzung der Anwendungsbeispiele

In den Anwendungsbeispielen wird die Lösung von Automatisierungsaufgaben im Zusammenspiel mehrerer Komponenten in Form von Text, Grafiken und/oder Software-Bausteinen beispielhaft dargestellt. Die Anwendungsbeispiele sind ein kostenloser Service der Siemens AG und/oder einer Tochtergesellschaft der Siemens AG ("Siemens"). Sie sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit und Funktionsfähigkeit hinsichtlich Konfiguration und Ausstattung. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern bieten lediglich Hilfestellung bei typischen Aufgabenstellungen. Sie sind selbst für den sachgemäßen und sicheren Betrieb der Produkte innerhalb der geltenden Vorschriften verantwortlich und müssen dazu die Funktion des jeweiligen Anwendungsbeispiels überprüfen und auf Ihre Anlage individuell anpassen.

Sie erhalten von Siemens das nicht ausschließliche, nicht unterlizenzierbare und nicht übertragbare Recht, die Anwendungsbeispiele durch fachlich geschultes Personal zu nutzen. Jede Änderung an den Anwendungsbeispielen erfolgt auf Ihre Verantwortung. Die Weitergabe an Dritte oder Vervielfältigung der Anwendungsbeispiele oder von Auszügen daraus ist nur in Kombination mit Ihren eigenen Produkten gestattet. Die Anwendungsbeispiele unterliegen nicht zwingend den üblichen Tests und Qualitätsprüfungen eines kostenpflichtigen Produkts, können Funktions- und Leistungsmängel enthalten und mit Fehlern behaftet sein. Sie sind verpflichtet, die Nutzung so zu gestalten, dass eventuelle Fehlfunktionen nicht zu Sachschäden oder der Verletzung von Personen führen.

Haftungsausschluss

Siemens schließt seine Haftung, gleich aus welchem Rechtsgrund, insbesondere für die Verwendbarkeit, Verfügbarkeit, Vollständigkeit und Mangelfreiheit der Anwendungsbeispiele, sowie dazugehöriger Hinweise, Projektierungs- und Leistungsdaten und dadurch verursachte Schäden aus. Dies gilt nicht, soweit Siemens zwingend haftet, z.B. nach dem Produkthaftungsgesetz, in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der schuldhaften Verletzung des Lebens, des Körpers oder der Gesundheit, bei Nichteinhaltung einer übernommenen Garantie, wegen des arglistigen Verschweigens eines Mangels oder wegen der schuldhaften Verletzung wesentlicher Vertragspflichten. Der Schadensersatzanspruch für die Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegen oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist mit den vorstehenden Regelungen nicht verbunden. Von in diesem Zusammenhang bestehenden oder entstehenden Ansprüchen Dritter stellen Sie Siemens frei, soweit Siemens nicht gesetzlich zwingend haftet.

Durch Nutzung der Anwendungsbeispiele erkennen Sie an, dass Siemens über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden kann.

Weitere Hinweise

Siemens behält sich das Recht vor, Änderungen an den Anwendungsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in den Anwendungsbeispielen und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Ergänzend gelten die Siemens Nutzungsbedingungen (https://support.industry.siemens.com).

Securityhinweise

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 Netzwerk-segmentierung) ergriffen wurden.

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

Inhaltsverzeichnis

Recl	Rechtliche Hinweise 2						
1	Aufgab	e	4				
	1.1 1.2	Übersicht Anforderungen	4 5				
2	Lösung]	6				
	2.1 2.2 2.3 2.4	Übersicht Gesamtlösung Beschreibung der Kernfunktionalität Verwendete Hard- und Software-Komponenten Alternativlösungen	6 7 9 11				
3	Grundla	agen	13				
	3.1 3.2 3.3 3.4 3.5 3.5.1 3.5.2	Prinzipielles Anwendungsmodell der OPC-DA-Schnittstelle Unterschiede zwischen synchronen und asynchronen Lese- und Schreibaufträgen Aufteilung von OPC-Items auf OPC-Gruppen Identifizierung von angelegten OPC-Items in einem OPC-Client Funktionsmechanismen von .NET und Einbinden der Komponenten der bisherigen Programmierwelt Programmiermodell der alten Welt Programmiermodell der .NET Welt	13 15 19 20 22 22 23				
	3.5.3	Integration von COM-Komponenten in .NET-Anwendungen	24				
_	3.6	Grundlagen zum Thema S7 Kommunikation	26				
4	Funktic	onsmechanismen dieser Applikation	29				
	4.1 4.2 4.3	COMDA Client API Simple Client COM DA S7 Programm	31 33 36				
5	Konfigu	uration und Projektierung des OPC-Servers	40				
	5.1 5.2 5.3	Projektierung des OPC-Servers in STEP7 V15.1 Projektierung der S7 Verbindungen Kontrollieren der Einstellungen	40 45 50				
6	Installa	tion und Inbetriebnahme	52				
	6.1 6.2 6.3 6.4 6.5	Installation der Hard- und Software Laden der der PC Station über STEP 7 V1x Importieren der XDB-Datei in den Komponenten Konfigurator Installation des OPC-Clients auf dem PC/PG Laden der Simulation auf die S7-Stationen	52 54 57 60 61				
7	Bedien	ung der Applikation	62				
	7.1 7.2	Übersicht Verwenden der Blockdienste	62 64				
8	Glossa	r	65				
9	Literatu	ırhinweise	66				
	9.1 9.2	Literaturangaben Internet-Link-Angaben	66 67				
10	Historie	9	67				

1 Aufgabe

1.1 Übersicht

Einleitung

Dieses Applikationsbeispiel zeigt exemplarisch die Kopplung eines Produktionsprozesses an einen Windows basierten PC mit einem Datenaustausch über OPC. Mit diesem Wirkungsprinzip können beispielsweise eigene, spezialisierte Bedienoberflächen und Prozessvisualisierung oder Datenerfassung realisiert werden.

Überblick über die Automatisierungsaufgabe

Folgendes Bild gibt einen Überblick über die Automatisierungsaufgabe. Abbildung 1-1



Beschreibung der Automatisierungsaufgabe

Ein Prozess wird mit zwei verschiedenen SPS simuliert. Die Daten aus diesem Prozess sollen angezeigt, aufgezeichnet und modifiziert werden können. Diese Applikation zeigt, wie — unter zu Hilfenahme der .NET Programmierumgebung – Prozessdaten auf einem PC angezeigt werden können unabhängig davon aus welcher Prozesssteuerung sie stammen.

1.2 Anforderungen

Anforderungen an die Bedien- und Beobachtungssoftware zur Visualisierung

Die Software soll schnelle und einfache Oberflächenerstellung ermöglichen. Dazu muss sie folgenden Anforderungen entsprechen:

- Erweiterbare Bibliothek mit Oberflächencontrols.
- Verwendung von Windows Standard Controls.
- Einfache, wieder verwendbare Anbindung dieser Controls an die Daten.

Anforderung an die Datenschnittstelle zwischen Visualisierung und Steuerung

- Anbindung an die Prozessdaten über Industrial Ethernet sowie den SIMATIC NET OPC-Server V8.x.
- Nutzung des OPC-DataAccess Custom-Interfaces über RCW.
- Symbolische und absolute Adressierung der Prozessdaten.
- Asynchrones/Synchrones Lesen und Schreiben von einzelnen Prozessdaten.
- Schreiben und Lesen von großen Datenmengen über Blockdienste.
- Implementierung eines Gerüstes zur Fehlerbehandlung.

Anforderungen an die einzusetzende Entwicklungsumgebung

Es soll die aktuelle Windows-Entwicklungsumgebung eingesetzt werden:

- Einsatz des Microsoft Visual Studio® .NET 2010 SP1.
- Einsatz der .NET-Programmiersprache Visual C#.

2 Lösung

2.1 Übersicht Gesamtlösung

Schema

Die folgende Abbildung zeigt schematisch die wichtigsten Komponenten der Lösung:

Abbildung 2-1



S7-Station

Die Steuerungsseite besteht aus zwei S7-Stationen: Einer CPU 315-2 PN/DP $\,$ und einer S7-1214C

PC Station

Eine PC-Station wird über eine handelsübliche Ethernet-Netzwerkkarte an eine S7-300 Steuerung und eine S7-1200 angeschlossen. Auf der PC-Station läuft der SIMATIC NET OPC Server und der OPC-Client. Ein sehr einfach gestalteter Client (Simple OPC Client) zeigt Ihnen alle Basisfunktionen für den Einstieg.

Die Funktionalität dieser Beispiel-Clients wird im nächsten Abschnitt erläutert.

Vorteile

Die vorliegende Applikation bietet Ihnen folgende Vorteile:

- Ein vollständiges C# Programmierbeispiel, das Ihnen alle wichtigen OPC-Mechanismen zeigt.
- Eine wieder verwendbare, erweiterbare Klassenbibliothek in denen die wichtigsten OPC-Methoden für eigene Anwendungen gekapselt sind.
- eine Vereinfachung der allgemeinen ".NET RCW" der OPC Foundation auf eine einfach verwendbare und erweiterbare Bibliothek für ".NET".

Abgrenzung

Diese Applikation enthält keine vollständige Beschreibung der folgenden Themen:

- ".NET"-Framework
- C#
- OPC-Spezifikation
- tiefgehende COM-Mechanismen

Grundlegende Kenntnisse über diese Themen werden vorausgesetzt.

Vorausgesetzte Kenntnisse

Grundlegende Kenntnisse im Bereich der objektorientierten Programmierung sowie im COM-Umfeld werden vorausgesetzt. Des Weiteren sind Kenntnisse in UML (Unified Modelling Language) von Vorteil.

2.2 Beschreibung der Kernfunktionalität

Beteiligte Software-Komponenten

Folgendes Bild zeigt die beteiligten Software-Komponenten. Die genauere Erläuterung erfolgt in Kap .4.

Abbildung 2-2



PC/PG

Auf dem PC/PG ist zur Visualisierung ein C# OPC-Client realisiert.

Der OPC-Client nutzt zur Ankopplung an den Prozess den OPC Runtime Callable Wrapper (kurz: RCW) der mit dem SIMATIC NET OPC-Servers automatisch mitinstalliert wurde.

Über die SIMATIC NET SOFTNET-S7-Anbindung und das S7-Protokoll baut der SIMATIC NET OPC-Server die Verbindung zur Steuerung auf.

Steuerung

Die Steuerung liefert die zu visualisierenden Daten.

Dazu ist ein einfaches S7-Programm zur Simulation der verschiedenen Datentypen implementiert.

Erstellte Softwarekomponenten

- C# OPC-Client
- STEP7-Simulationsprogramm

Allgemeiner Ablauf der Applikation

Der OPC-SimpleClient zeigt den Zugriff auf OPC Datenpunkte und die Verwendung der Variablen sowie der Blockdienste des OPC-Servers. Die Client Bibliothek ist erweiterbar:

Tabelle 2-1

Aktion	Oberfläche
 Folgende Funktionen sind im OPC-Sim Anbindung an den OPC-Server: Verbinden mit dem OPC-Server. Anlegen von Gruppen und Items. Werte Beobachten. Werte Lesen/Schreiben. Black Lesen/Schreiben. 	pleClient implementiert:
Simple Client OPC COM DA Connect opcda://localhost/Opc.SimaticNet S7 300 - using items on an S7 300 device. Block Read / Wi S7 1200 - using items on an S7 1200 device. Block Read /	OPC DA Server URL rite is enabled Write is disabled
ItemID Monitored Value Dynamic/Analog Types/Double Static/Simple Types/Int Monitor	le Read Value Write Value
Block Read ItemID Block Read Static/ArrayTypes/Byte[] Monitor Block	Block Read Result
Block Write ItemID Block Write Static/ArrayTypes/Byte[] 4096 Write	te Length in Byte Write Block 1 Write [0, 1, 2 255] Write Block 2 Write [255, 254,

Hinweis

Die Aktionen werden über die Bedienelemente der Oberfläche durchgeführt.

2.3 Verwendete Hard- und Software-Komponenten

Die Applikation wurde mit den nachfolgenden Komponenten erstellt:

Hardware-Komponenten für die Steuerung

Tabelle 2-2					
Komponente	Anz.	MLFB/Bestellnummer	Hinweis		
PS307 5A	1	6ES7307-1EA00-0AA0			
CPU 315-2 PN/DP	1	6ES7315-2EH14-0AB0	Oder eine vergleichbare S7-300 CPU.		
S7-1200 PM 1207	1	6EP1332-1SH71			
CPU 1214C DC/DC/DC	1	6ES7 214-1AE30-0XB0	Oder eine vergleichbare S7- 1200 CPU.		
Handelsüblicher Switch	1	Abhängig vom Produkt			

Hardware-Komponenten für den PC

Tabelle 2-3

Komponente	Anz.	MLFB/Bestellnummer	Hinweis
Power PG	1	6ES7751-0EA31-0LB3	Oder ähnlicher PC/PG.
NDIS-fähige Netzwerkkarte	1	Abhängig vom Produkt	Im Power PG integriert.

Standard Software-Komponenten

Tabelle 2-4

Komponente Anz.		Bestellnummer	Hinweis	
STEP 7 Professional V15.1	1	6ES7822-4AA03-0YA5		
WinCC Professional V15.1	1	6AV2103-0DA00-0AM0	Passen Sie die Bestellnummer entsprechend der nötigen Powertags an.	
SIMATIC NET IE SOFTNET-S7 (V12)	1	6GK1704-1LW12-0AA0 6GK1704-1CW12-0AA0	LW=8 S7-Verbindungen (Lean), CW=64 S7-Verbindungen	
Microsoft Visual Studio 2010	1	Express Edition Standard Edition Professional Edition	Erhältlich im Microsoft Store (http://emea.microsoftstore.com)	
.NET Framework 3.5	1	Frei downloadbar bei http://www.microsoft.com/	Wird durch SIMATIC NET installiert	

Beispieldateien und Projekte

Die folgende Liste enthält alle Dateien und Projekte, die in diesem Beispiel verwendet werden.

Tabelle 2-5

Komponente	Hinweis
21043779_OPCClient_RCW_CODE.zip	Die Datei enthält das archivierte STEP7-Simulationsprogramm für STEP7 V11 und STEP7 V13, den Setup zur Installation der OPC-Clients (ohne .NET Framework) und die vollständigen Source-Code Dateien
21043779_OPCClient_RCW_DOKU_V2_1_d.pdf	Dieses Dokument

2.4 Alternativlösungen

OPC Lösungen in verschiedenen Programmiersprachen

Zur Kopplung von Prozess und Bedienen & Beobachten wird OPC als Standardmechanismus eingesetzt. Entsprechend der bevorzugten Programmiersprache gibt es verschiedene Lösungen, die sich vor allem hinsichtlich Geschwindigkeit der Programmierung, Art der Zielapplikation und deren Vorraussetzungen unterscheiden. Weiterhin gilt es zu unterscheiden, welche der OPC-Schnittstellen verwendet werden soll: die klassischen COM basierten Schnittstellen OPC DA2, DA3, A&E oder die kombinierte OPC UA Schnittstelle.

Entscheidungskriterien für die Nutzung verschiedener Programmiersprachen

Die folgende Tabelle zeigt Ihnen die wichtigsten Entscheidungskriterien, die ausschlaggebend sind für die Wahl der einzusetzenden Programmiersprache: Tabelle 2-6

Entscheidungskriterium	Beschreibung		
Custom-Interface	Ist die Nutzung des Custom-Interfaces der OPC DA- Schnittstelle möglich?		
Automation-Interface	Ist die Nutzung des Automation-Interfaces der OPC DA- Schnittstelle möglich?		
Einfache Implementierung	Inwieweit ist die Sprache geeignet, Code relativ einfach zu implementieren?		
Fehlerhandling	Wie gut ist die Sprache geeignet, ein professionelles Fehlerhandling zu realisieren?		
Performance	Wie gut eignet sich die Sprache, um performante OPC- Anwendungen zu entwickeln?		
Parallelität	Wie gut eignet sich die Sprache, um parallele Tasks (aus verschiedenen Threads) zu bearbeiten?		

Gegenüberstellung verschiedener Programmiersprachen

In der nachfolgenden Tabelle werden nun die verschiedenen Programmiersprachen anhand der oben beschriebenen Kriterien gegenübergestellt. Die in diesem Beispiel realisierte Variante ist in der Tabelle blau hinterlegt.

Tabelle 2-7

Kriterium	VBA (MSOffice)	VB V6.0	VC++	.NET-Sprachen(managed)
Custom-Interface	-	-	~	✓ (mit RCW)
Automation-Interface	\checkmark	✓	√ (*)	✓ (*) (mit RCW)
Einfache Implementierung	++	++	+ (**)	++
Fehlerhandling	-	-	++	++
Performance	-	-	++	+
Parallelität	-	-	++	+

(*) sollte nicht verwendet werden

(**) unter Verwendung der Active Template Libary (ATL)

Entscheidungskriterien für die Nutzung OPC Schnittstellen

Die folgende Tabelle zeigt Ihnen die wichtigsten Entscheidungskriterien, die ausschlaggebend sind für die Wahl der einzusetzenden OPC-Schnittstelle: Tabelle 2-8

Entscheidungskriterium	Beschreibung
COM/DCOM OPC DA 2/3	Ist die Nutzung OPC DA-Schnittstelle mit Lesen / Schreiben / Beobachten ausreichend für die Anwendung?
COM/DCOM OPC A&E	Wird (zukünftig) auch eine Verarbeitung von Events und Prozess-Alarmen angestrebt?
OPC UA	Wird eine kombinierte Funktion (Lesen / Schreiben / Alarme / Methoden / Objekte / Typen) mit erweiterten Funktionen benötigt?
Sicherheit	Sollen remote Verbindungen aufgebaut und sicher betrieben werden, über Firewallgrenzen hinweg?
Plattformen	Soll die Applikation nur mit windowsbasierten Systemen kommunizieren können (kein Linux, Android, iOS)?
Funktionalität, Flexibilität	Welche Funktionalität wird vom Interface bereitgestellt? Inwieweit sollen Funktionen ergänzt werden können?
Kombination mit anderen Herstellern	Wie gut eignet sich die Schnittstelle, um die Systeme anderer Hersteller einzubinden, was bleibt erhalten was sollte konfigurierbar sein?

Gegenüberstellung verschiedener OPC Bibliotheken für .NET Sprachen

Um die Implementierung von OPC Client Anwendungen in den .NET Sprachen zu erleichtern, gibt es verschiedene Bibliotheken für unterschiedliche Einsatzbereiche. In der nachfolgenden Tabelle werden die Bibliotheken Anhand der oben beschriebenen Kriterien gegenübergestellt.

Tabelle 2-9

Kriterium	OPC Connector	OPCClient.API	OPCDa.RCW	OPCUa.RCW
COM DA-Interface	\checkmark	✓	✓	-
COM AE-Interface	-	-	-	-
OPC UA-Interface	✓ (*)	✓ (*)	-	✓
Sicherheit	- (**)	- (**)	- (**)	++
Plattformen	-	-	-	✓
Funktionalität, Flexibilität	-	+	++	+++
andere Hersteller	- (***)	- (***)	✓	✓

(*) nur DA Funktionen

(**) Sicherheit nur über DCOM

(***) klassisch OPC DA und OPC UA parallel, limitiert auf SimaticNET OPC Server

3 Grundlagen

3.1 Prinzipielles Anwendungsmodell der OPC-DA-Schnittstelle

Einleitung

In diesem Abschnitt wird kurz auf das Anwendungsmodell der OPC DA-Schnittstelle eingegangen. Eine ausführliche Beschreibung des Objektmodells finden Sie in der OPC DA-Spezifikation der OPC Foundation.

OPC-Client

Indem der OPC-Client Referenzen auf die jeweiligen Schnittstellen erzeugt, teilt der OPC-Client dem OPC-Server mit, dass er auf bestimmte Variablen Zugriff wünscht.

Weitere über die Group(s)-Schnittstelle "IOPCGroupStateMgt" verfügbare Schnittstellen erlauben lesende und schreibende Zugriffe auf diese Variablen. Folgende Zugriffsverfahren sind dabei möglich:

- Synchron lesen / schreiben.
- Asynchron lesen / schreiben.
- Beobachten von Variablen (Melden von Wertänderungen).

Hinweis Single-Threaded OPC-Clients werden bei synchronen Lese- / Schreibaufträgen blockiert. Bei asynchronen Lese- / Schreibaufträgen bleibt der OPC-Client bedienbar ("responsive").

Für weitere Informationen siehe Kap. 3.2.

OPC-Server

Der OPC-Server stellt die zentrale Kommunikationseinheit zwischen einem OPC-Client und der jeweiligen Steuerung dar.

Über COM/DCOM-Mechanismen stellt er zum OPC-Client hin genormte Schnittstellen zur Verfügung, die es jeder COM-fähigen Anwendung erlauben, auf Variablen einer beliebigen Steuerung zuzugreifen.

Dabei sind in dieser Applikation die DataAccess-Schnittstellen zur Erstellung der logischen Verknüpfungen zwischen OPC-Items und Prozessvariablen ausschlaggebend.

Die Anbindung des OPC-Servers an die jeweilige Steuerung erfolgt über die implementierten Kommunikationsprotokolle.

Verbindung zwischen OPC-Server und Steuerung

Über die im OPC-Server projektierten Verbindungen werden mithilfe des jeweiligen Protokolls die Werte der Variablen ausgelesen. In diesem Beispiel wird das S7 Protokoll verwendet, das zwei prinzipielle Übertragungsmechanismen beinhaltet, die Variablendienste und die Blockdienste. Die Verbindungen werden im TIA-Portal projektiert (siehe Kap.5.2).

Auslesen der Variablen aus der Steuerung

Sie können die Variablen auf zwei verschiedene Arten Auslesen und Aktualisieren: Tabelle 3-1

Dienst	Beschreibung
Variablendienst	Bei Variablendiensten werden eine oder mehrere Prozessvariablen durch absolute oder symbolische Adressierung angegeben. Dabei können die zu beobachtenden Variablen zyklisch vom OPC-Server abgefragt werden (→Polling, s. Glossar). In der Steuerung muss hier nichts programmiert werden. Die Kommunikation erfolgt über systeminterne Prozesse.
Blockdienst	Von Blockdiensten spricht man, wenn die zu beobachtenden Variablen programmgesteuert mittels größerer Datenblöcke zum OPC-Server übertragen werden. Die Steuerung ruft in Ihrem Ablaufprogramm einen Kommunikationsbaustein (BSEND) auf, um aktiv die Datenübertragung anzustoßen. Der OPC-Server empfängt die Daten in seinem Empfangspuffer. Dabei werden nicht die Prozessvariablen an sich adressiert, sondern vielmehr die Datenbereiche, wie z. B. ein Datenbaustein. Ein zyklisches "Beobachten" des Empfangspuffers seitens des OPC- Servers ist hierbei nur für Empfangs-Items (z. B. "receive") sinnvoll.

Hinweise zu den Diensten

- In dieser Applikation werden Variablendienste und Blockdienste (nur S7-300) benutzt.
- Die absolute Adressierung unterscheidet sich von der symbolischen Adressierung nur durch die ITEM_ID.
- Die Unterscheidung zwischen Variablen- und Blockdiensten erfolgt seitens des OPC-Clients nur durch dessen ITEM_ID.
- Mit "Blockdiensten" sind hier ausschließlich die durch den SIMATIC NET OPC Server zur Verfügung gestellten Dienste ("SEND/RECEIVE", "BSEND/BRECEIVE") gemeint.

3.2 Unterschiede zwischen synchronen und asynchronen Lese- und Schreibaufträgen

Im Folgenden gehen wir auf die Unterschiede zwischen synchronem und asynchronem Lesen und Schreiben von OPC-Items ein. Im Speziellen erläutern wir die Unterschiede zwischen Zugriffen auf das "DEVICE" und den "CACHE".

Wirkungsweise von synchronen/asynchronen Lese- und Schreibaufträgen

Folgendes Bild gibt Ihnen einen Überblick über die Wirkungsweise der jeweiligen Aufrufe. Die Wirkungsweise ist mit einer CPU mit PROFINET-Schnittstelle identisch.



Abbildung 3-1

Erläuterung zum Bild

- Die roten Pfeile zeigen die Aufrufrichtung, die blauen Pfeile den resultierenden Datenfluss.
- Die Daten zwischen dem OPC-Server und der Steuerung werden über eine Industrial Ethernet-Karte und der internen Schnittstelle einer PN-CPU oder einen CP343-1 ausgetauscht.
- Der Datenaustausch zwischen OPC-Client und OPC-Server findet als Interprozess-Kommunikation statt (hier: COM).
- Der "Cache" bezeichnet einen Zwischenpuffer, den der OPC-Server für eine bestimmte Gruppe einrichtet. Er enthält ein lokales Abbild der für diese Gruppe definierten Prozessvariablen (die von der Gruppe wiederum als OPC-Items verwaltet werden).
- Das "Device" bezeichnet die Prozessvariablen auf der Steuerung.

Erläuterung der Vorgänge

Die nun folgende Tabelle erklärt die jeweiligen Aufrufe und Datenflüsse. Dabei wird auch auf die jeweiligen Folgen hingewiesen:

Тэ	ho	ماا	3.	2
ıа	be	ne	3	·2

Nr.	Aktion	Anmerkung	
1	Synchroner Leseauftrag auf den CACHE	Der Wert eines OPC-Items wird mit dem aktuell im CACHE vorliegenden Wert ausgelesen.	
2	Aktualisierung des CACHEs	Der OPC-Server aktualisiert nach der mit "requestedUpdateRate" festgelegten Zeitdauer den gesamten CACHE mit den Werten aus dem DEVICE. Hinweis:	
		Die requestedUpdateRate bzw. auch die vom OPC-Server zurückgelieferte Aktualisierungszeit ("revisedUpdateRate") muss nicht mit der tatsächlichen Aktualisierungszeit übereinstimmen. Beachten Sie hierbei, dass die tatsächliche Aktualisierungszeit keinesfalls kürzer ist als die revisedUpdateRate.	
3	OnDataChange-Event	Der OPC-Server hat bei OPC-Items, die als "aktiv" parametriert sind, eine Datenänderung innerhalb seines CACHES registriert (neuer gepollter Wert <> gecachter Wert). Diese Änderung wird dem OPC- Client gemeldet und der neue Wert wird in den CACHE übernommen. Hinweis:	
		Dieser Mechanismus ist optimal, um Prozessvariablen zu beobachten. Beachten Sie hierbei dass neben einer Werteänderung auch eine Änderung des Status einen DataChange-Event auslöst. Die Aktualisierungszeit entspricht der tatsächlichen "updateRate". Der DataChange- Event informiert lediglich über die geänderten Items.	
4	Asynchroner Leseauftrag auf das DEVICE	Der OPC-Client startet über die OPC-Gruppe einen asynchronen Leseauftrag. Der OPC-Server liest die angeforderten Prozessvariablen aus dem Prozessabbild der S7-CPU und liefert diese über das OnReadComplete-Ereignis dem OPC-Client. Dabei schreibt er die gelesenen Werte in seinen CACHE. Hinweis: Der OPC-Client ist bei asynchronen Aufrufen, während der jeweilige Auftrag läuft, weiter bedienbar ("responsive").	
5	Synchroner Leseauftrag auf das DEVICE	Der Wert eines OPC-Items wird mit dem aktuell im Prozessabbild der S7-CPU vorliegenden Wert ausgelesen. Hinweis:	
		Dieser Zugriff kann je nach Physik, Datenverkehr und Datenmenge mehrere Sekunden dauern.	
		In dieser Zeit kann der Thread des OPC-Client, der den synchronen Leseaufruf startete, keine anderen Aufgaben durchführen! Single-threaded OPC-Clients sind also während dieser Zeit blockiert und können keine weiteren Bedien- und Beobachtaufgaben ausführen!	

6	Synchroner oder asynchroner Schreibauftrag	Schreibaufträge werden, unabhängig davon ob diese synchron oder asynchron sind, nur auf das DEVICE ausgeführt. Hinweis:
		Dieser Zugriff kann je nach Physik und Datenmenge mehrere Sekunden dauern.
		Wurde ein synchroner Schreibauftrag abgesetzt, so tritt das gleiche Verhalten auf, wie in Punkt 5 beschrieben; bei einem asynchronen Aufruf verhält sich der OPC-Client wie in Punkt 4 beschrieben.

Zugriff auf CACHE und DEVICE

Folgende Tabelle zeigt zusammenfassend, welche Operationen auf das "DEVICE" und den "CACHE" möglich sind:

Tabelle 3-3				
Operation	DEVICE	CACHE	Hinweis	
Synchrones Lesen	✓	✓	Das Lesen vom CACHE erfordert, dass die Gruppe und das jeweilige Item aktiv sind.	
Asynchrones Lesen	~	-	Der OPC-Server liest die Variable vom DEVICE. Dabei wird auch dessen CACHE aktualisiert.	
Synchrones Schreiben	~	-	Schreibaufträge werden immer auf das DEVICE geschrieben.	
Asynchrones Schreiben	~	-	Schreibaufträge werden immer auf das DEVICE geschrieben.	
Beobachten (ereignisgesteuert)	-	✓	Das Beobachten von Variablen ist ausschließlich unter Verwendung des CACHE möglich.	

Hinweis Neben Lese- und Schreibaufträgen sind noch weitere Methoden verfügbar (wie z. B. IOPCAsyncIO2::Refresh), die jedoch in dieser Applikation nicht eingesetzt wurden.

Für weitere Informationen s. OPC-Spezifikation der OPC Foundation.

Gegenüberstellung der Anwendungsfälle

Folgende Tabelle enthält eine Gegenüberstellung verschiedener Anwendungsfälle und zeigt das empfohlene Vorgehen:

Tabelle 3-4

Operation	Zyklisch/ Azyklisch	Datenmenge	DataChange- Mechanismus	Sync	Async
	Zyklisch beobachten	Groß	+	*	*
Lagan		Klein	+	*	*
Lesen	Azyklisch	Groß	-	-	+
		Klein	-	+	-

Schreiben	Azyklisch	Groß	-	-	+
		Klein	-	+	-

(*) Der OnDataChange-Mechanismus wird asynchron auf den CACHE ausgeführt

Hinweis Große Datenmengen im Bereich mehrerer Kilobyte sollten mit den Blockdiensten, z. B. "BSEND/BRCV", gelesen bzw. geschrieben werden. Hierbei agiert die SIMATIC als aktiver Kommunikationspartner.

3.3 Aufteilung von OPC-Items auf OPC-Gruppen

Im Folgenden erfahren Sie, welche Kriterien zur Aufteilung der OPC-Items auf OPC-Gruppen berücksichtigt werden sollten.

Motivation der Aufteilung von OPC-Items

Wie im Kap. 3.2 erläutert, resultieren aus bestimmten Aufruf-Typen bestimmte Verhaltensweisen.

Nun ist es sinnvoll, Prozessvariablen bzw. OPC-Items, die ein ähnliches Bedienund Beobachtverhalten erhalten sollen, in Gruppen zusammenzufassen.

In manchen Fällen ist es jedoch erwünscht, dass eine Prozessvariable z. B. an zwei unterschiedlichen Stellen des OPC-Clients beobachtet werden soll (wie z. B. in zwei unterschiedlichen Seiten oder Dialogen). Ein weiteres Beispiel wäre, dass eine Prozessvariable an einer Stelle beobachtet, an einer anderen Stelle jedoch nur sporadisch (also azyklisch) ausgelesen werden soll.

In solchen Fällen ist es sinnvoll, ein und dieselbe Prozessvariable in mehreren Gruppen anzulegen.

Aufteilung von Prozessvariablen auf OPC-Items und OPC-Gruppen

Folgendes Bild veranschaulicht diese Möglichkeit mit zwei unterschiedlichen Gruppen. Hierbei wird die Prozessvariable 1 auf zwei OPC-Items abgebildet, die in zwei Gruppen hinterlegt ist. Somit können für eine Prozessvariable unterschiedliche Verhaltensweisen in einem OPC-Client erzielt werden.

Abbildung 3-2



3.4 Identifizierung von angelegten OPC-Items in einem OPC-Client

Damit die auf OPC-Items abgebildeten Prozessvariablen von einem OPC-Client gelesen bzw. geschrieben werden können, muss eine eindeutige Zuordnung der OPC-Items zu einem OPC-Client existieren.

Hierfür wird das Konzept von Identifikatoren oder "Handles" eingesetzt. Es ermöglicht eine effizientere Übertragung und schnellere Identifizierung der Items bei Zugriffsoperationen.

Handle-Typen

Man unterscheidet zwischen zwei Handle-Typen. Durch diese Unterscheidung kann garantiert werden, dass sowohl der OPC-Client als auch der OPC-Server die jeweiligen OPC-Items eindeutig identifizieren können.

Tabelle 3-5

Тур	Beschreibung
Client handles	Diese Handles werden vom Client erzeugt und dienen zur Identifikation eines OPC-Items innerhalb des OPC-Clients. Beim Anlegen der Items (AddItems) teilt der Client dem Server seine Client- Handles mit.
Serverhandles	Diese Handles werden vom Server erzeugt und dienen zur Identifikation eines OPC-Items innerhalb des OPC-Servers. Beim Anlegen der Items (AddItems) gibt der Server dem Client seine Server-Handles zurück.

Serverhandles

Ist die Aufrufrichtung vom OPC-Client zum OPC-Server (z. B. Write), so muss der OPC-Client dem OPC-Server die jeweiligen Serverhandles übergeben.

Beispiel: Schreibe in die OPC-Items!

Abbildung 3-3



Clienthandles

Ist die Aufrufrichtung vom OPC-Server zum OPC-Client (z. B. OnDataChange), so werden dem OPC-Client vom OPC-Server die jeweiligen Clienthandles übergeben.

Beispiel: Die OPC-Items im CACHE haben sich verändert. \rightarrow Es wird das OnDataChange-Ereignis ausgelöst.





Auswirkung auf das Client-Programm

Diese Struktur impliziert, dass ein OPC-Client, je nach Anwendungsfall, die Serverund/oder die Clienthandles verwalten muss.

So können z. B. die Clienthandles als Index des jeweiligen OPC-Items in einem OPC-Item-Array dienen.

Die Serverhandles sollten entweder in einem separaten Serverhandle-Array oder in einer eigenen OPC-Item-Verwaltungsklasse gekapselt sein.

3.5 Funktionsmechanismen von .NET und Einbinden der Komponenten der bisherigen Programmierwelt

Es wird nun zunächst kurz auf die bisherige sowie die neue Windows-Programmierwelt eingegangen. Anschließend werden die zu beachtenden Unterschiede behandelt.

Es handelt sich hierbei lediglich um eine kurze Einführung, die für das Verständnis des "wie programmiert man mit .NET" und des "wie bringt man die neue und die alte Welt zusammen?" hilfreich ist. Für weitere Informationen sei hier auf Sekundärliteratur verwiesen.

3.5.1 Programmiermodell der alten Welt

Übersicht



API

Das bisherige Programmiermodell der Windows-Welt baut auf ein **A**pplication **P**rogrammers Interface (API) auf. Die API stellt eine Menge von C-Funktionen für den Zugriff auf Betriebssystem-Ressourcen und -Funktionalitäten zur Verfügung.

Programmiersprachen und Bibliotheken

Mit verschiedenen Programmiersprachen wurde es mit teilweise umfangreichen Bibliotheken (**VBRun**time, **M**icrosoft Foundation **C**lasses, **A**ctive **T**emplate Library) ermöglicht, Windows-Programme zu implementieren. Dabei kapselten die Bibliotheken jeweils einen Teil der API-Funktionen.

Siemens AG 2019 All rights reserved

Zusammenarbeit der Sprachen mit COM

Damit unterschiedlich entwickelte Programmkomponenten miteinander agieren konnten, wurde von Microsoft das **C**omponent **O**bject **M**odell (COM) eingeführt. Damit war es unter Verwendung einheitlicher Schnittstellendefinitionen möglich, einen "connecting glue" zwischen den einzelnen Komponenten aufzubauen. Um die sprachen- und komponentenübergreifende Interaktionen auf rechnerübergreifende Interaktionen zu erweitern, wurde das Konzept des **D**istributed **C**omponent **O**bject **M**odell (DCOM) entwickelt.

Hinweis Der OPC-Server arbeitet als COM-Komponente.

→ OPC-Clients greifen über COM-Mechanismen auf den OPC-Server zu.

3.5.2 Programmiermodell der .NET Welt

Übersicht



API

Basis des Modells ist, wie auch in der bisherigen Microsoft Programmierwelt, die Win32 API.

CLR

Auf der API aufsetzend folgt die Common Language Runtime (CLR). Sie stellt eine Laufzeitumgebung dar, deren Modell mit einer Java Virtual Machine vergleichbar ist. Sie übersetzt Just In Time (JIT) eine Art Bytecode in X86-Code (dieser ist von der jeweiligen Prozessorarchitektur abhängig); Microsoft spricht hierbei von IL-Code (Intermediate Language).

Intermediate Language, Garbage Collector und unmanaged Code

Vorteil des IL-Codes ist, dass er plattformunabhängig ist. Es ist so z. B. auch möglich, diesen Code auf Linux (Unix)-Systemen auszuführen, sofern dort eine CLR implementiert ist (siehe \6\)

Die CLR übernimmt nicht nur das JIT-kompilieren des IL-Codes, sondern auch die gesamte Speicherverwaltung. So wurde, ähnlich wie in Java, ein **G**arbage **C**ollector (GC) entwickelt, der in der CLR nicht mehr referenzierte

Speicherbereiche nach einer nicht festgelegten Zeit freigibt. Code, der dem Zugriff des Garbage Collectors unterliegt, bezeichnet man auch als "managed Code".

Damit es in bestimmten Fällen dennoch möglich ist, die Freigabe von Speicherbereichen manuell zu kontrollieren, kann der entwickelte Code als "unmanaged" deklariert werden. Damit ist es möglich bestimmte Code-Bereiche oder ganze Programme explizit vor dem Zugriff des Garbage-Collectors zu schützen.

Base Classes

Was in der alten Welt die teilweise unterschiedlich mächtigen Bibliotheken angeht, hat Microsoft nun eine einheitliche Basis-Klassen-Bibliothek implementiert, auf die unabhängig von der .NET-Programmiersprache zugegriffen werden kann.

Integrator Visual Studio .NET

Das Visual Studio .NET (VS .NET) übernimmt hier die Aufgabe, die unterschiedlichen .NET-Sprachen untereinander zusammenzufassen.

So ist es ohne Weiteres möglich, .NET-Komponenten in unterschiedlichen Sprachen zu realisieren. Interaktionen zwischen den .NET-Komponenten sind mit Visual Studio .NET einfach möglich.

Somit ist vor allem durch die einfach handzuhabenden Integrationsmöglichkeiten innerhalb der .NET-Komponenten ein schnelleres Entwickeln möglich.

3.5.3 Integration von COM-Komponenten in .NET-Anwendungen

Übersicht

Nutzung einer COM-Komponente in einer .NET-Anwendung/Komponente: Abbildung 3-7



Runtime Callable Wrapper (RCW) und der Zusammenhang zur COM-Komponente

Damit eine für die bisherige Windows-Welt erstellte COM-Komponente in eine .NET-Anwendung eingebaut werden kann, muss ein "Wrapper", eine Art Hülle, erstellt werden, die alle Schnittstellendefinitionen für die .NET-Anwendung kapselt.

Dies ist notwendig, da die bisherigen Schnittstellendefinitionen, die sich in sogenannten IDL-Dateien (Interface Definition Language, s. Glossar) befinden, von .NET nicht mehr unterstützt werden. Dieser Wrapper wird auch als Runtime Callable Wrapper (RCW) bezeichnet.

Bei COM-Komponenten, die das Automation Interface anbieten, kann Visual Studio diese Wrapper automatisch erstellen.

Für COM-Komponenten, die das Custom Interface zur Verfügung stellen, müssen solche Wrapper jedoch manuell erstellt werden.

Bei dem durch die OPC Foundation definierten Custom Interfaces (z. B. OPC DA 2 / 3) wurden die Wrapper durch die OPC Foundation manuell erstellt und allgemein zur Verfügung gestellt. Diese .NET OPC RCW werden z. B. mit der SIMATIC NET CD mitinstalliert.

Hinweis Die Unterschiede zwischen Automation und Custom Interface werden hier nicht weiter vertieft. Dazu ist Sekundärliteratur /2/ notwendig.

Speichermanagement bei Nutzung von COM-Komponenten

Das Speichermanagement wird bei .NET-Anwendungen vom GarbageCollector übernommen. Da COM-Komponenten jedoch ein explizites Speichermanagement verlangen, muss hier auf folgende Dinge für Datenaustausch zwischen beiden Komponenten geachtet werden:

Tabelle 3-6

Richtung	Beschreibung
.NET zu COM	In .NET sind alle Variablen Objekte. Da im allgemeinen COM-Server nicht synchron mit den COM-Clients laufen, sollten Übergabewerte vor dem Zugriff des GarbageCollectors geschützt werden ("anpinnen" von Objekten).
COM zu .NET	COM-Komponenten liefern Rückgabewerte in Form von COM-Pointern. Da es im managed Code von .NET-Clients keine solchen Pointer gibt, müssen die Rückgabewerte zunächst in .NET-Objekte gespeichert wer- den. Dies geschieht über den .NET-Datentyp "IntPtr" und Methoden / Objekten der "System.Marshal"-Klassen.

RCW und OPC

OPC-Komponenten bieten aufgrund der OPC-Spezifikation ein Custom Interface an, d. h. für performante Anwendungen muss ein RCW manuell erstellt werden.

Für den SIMATIC NET OPC-Server wird für die DataAccess-Schnittstelle V2.05 und die V3.0 ein RCW mitgeliefert, der in das jeweilige Visual Studio-Projekt integriert werden muss.

Wird hingegen das Automation Interface von OPC mit .NET genutzt, so wird der RCW automatisch vom VS .NET erstellt, sobald ein Verweis auf die OPC-Automation-Schnittstelle in das .NET-Projekt eingefügt wird. Durch die dadurch entstehende doppelte Kapselung des OPC-Custom-Interfaces ist allerdings ein Leistungsabfall in Kauf zu nehmen.

Hinweis Diese Applikation behandelt die Nutzung des Custom Interfaces, da eine performante Anbindung im Fokus der Anwendung steht.

Hinweis Der umgekehrte Anwendungsfall– die Nutzung einer .NET-Komponente als COM-Komponente – ist mit Hilfe eines Com Callable Wrappers (CCW) möglich. Im Rahmen dieses Dokuments wird jedoch nicht näher darauf eingegangen, da es zum Verständnis der Applikation nicht nötig ist.

Grundlagen zum Thema S7 Kommunikation 3.6

Der "klassische" OPC Server (COM/DCOM) ist für alle anderen Protokolle wie DP, FDL, SR oder SNMP vorhanden. Auf der SIMATIC NET CD V8.0 gibt es zusätzlich einen OPC UA Server. Der OPC UA Server ist nicht Bestandteil dieser Beschreibung.

Wirkungskette der Kommunikation

Die S7 Kommunikation wird in zwei grundlegend verschiedene Kommunikationsdienste unterteilt, in Variablendienste und Blockdienste. Diese sind auf der Ebene des OPC Servers fast vollständig verdeckt. Nur anhand der ItemId lässt sich erkennen, welcher Kommunikationsdienst zur Steuerung hin verwendet wird. Der Protokollpräfix "S7:" legt fest, dass es sich um die direkte Adressierungsart handelt. Der symbolische Zugriff verwendet den Präfix "SYM:".

Der OPC-Server zerlegt intern die ItemId in ihre Bestandteile und erkennt an ihrem Aufbau, über welchen Kommunikationsdienst zur S7 Steuerung kommuniziert werden muss. Hierbei identifiziert der Verbindungsname den Kommunikationspartner (dieser Name repräsentiert unter anderem z. B. eine IP Adresse) und das Schlüsselwort "BRCV" bzw. "BSEND" bewirkt die Verwendung der Blockdienste anstelle der Variablendienste. Der S7 Typbezeichner und die Offsetadresse geben die Position der Daten innerhalb der Steuerung an und der

Datentyp bestimmt die Interpretation dieser Daten. Abbilduna 3-8 **OPC Client** die Syntax der verwendeten ItemIDs entscheidet über den Dienst der zur Kommunikation über das S7 Protokoll verwendet wird



Variablendienste

Eine S7 Steuerung antwortet auf Anfragen über Variablendienste, hierzu ist nur eine einseitig projektierte Verbindung erforderlich. Jede S7-Steuerung ist ein so genannter "S7-Server" und beantwortet PUT/GET Anfragen, ohne dass dafür im Steuerungsprogramm der SPS etwas implementiert werden muss. Es kann direkt auf alle Datenbereiche der Steuerung zugegriffen werden (E, A, M, DB, etc.). Dieser Kommunikationsdienst ist sehr flexibel und vor allem einfach zu verwenden.

ItemIds für Variablendienste

S7:[<connectionname>]DB<no>,{<type>}<address>{,<quantity>}

Beispiel: S7:[S7-Verbindung_3]DB10,W20

Eine Variable vom Typ Wort (16Bit ohne Vorzeichen), die sich im Datenbaustein 10 befindet und an der Offset-Byteadresse 20 anfängt (also aus den Bytes 20 und 21 besteht). Diese Variable wird mit Put/Get über die Verbindung mit dem Namen "S7-Verbindung_3" geholt, also von der S7-Steuerung, die sich hinter dieser Verbindung verbirgt.

Symbolische ItemIDs

Neben der direkten Adressierung gibt es die Möglichkeit der symbolischen Adressierung. Hierzu wird von STEP7 aus (TIA Portal) der Adressraum generiert. Für alle symbolischen Bezeichner der Datenpunkte in den S7-Steuerungen, die über eine S7-Verbindung mit einem OPC-Server verbunden sind, kann ein Symbolexport angestoßen werden. Die daraus entstehende Symboldatei mit der Endung ATI wird dem OPC-Server per Download aus STEP7 oder per XDB-Import bekannt gemacht. Die ATI-Datei (Advanced Tag Information) beinhaltet eine Abbildung der symbolischen Namen auf die direkten Adressen.

Hinweis Alle Symbole werden letztendlich über PUT/GET von den Steuerungen geholt. Symbole, die eine BSEND bzw. BRCV Variable abbilden, sind über die Generierung von STEP7 nicht möglich.

Blockdienste

Für den Austausch großer Datenmengen steht der effektivere Blockdienst zur Verfügung. Auf einer beidseitig projektierten Verbindung können große Datenmengen (bis 64kByte) ausgetauscht werden. Die Kommunikation basiert auf dem Austausch von Datenpuffern. Allerdings müssen im Steuerungsprogramm hierzu entsprechende Systemfunktionsbausteine (BSEND/BRECV) aufgerufen werden. Der OPC-Server stellt auf dem PC die entsprechenden Gegenstücke bereit, wenn die dazugehörigen OPC-Items angelegt werden.

Aufbau der ItemIds für Blockdienste

S7:[<connectionname>]BRCV,<RID>{,{<type>}<address>{,<quantity>}}

Beispiel: S7:[S7-Verbindung_5]brcv,3

Der komplette Empfangspuffer für das BSEND/BRECV Paar mit der ID 3, dass über die Verbindung mit dem Namen "S7-Verbindung_5" angeschlossen ist, wird für OPC in einen Bytearray abgebildet. Dieses Bytearray enthält immer die zuletzt vom Kommunikationspartner (auf der anderen Seite von "S7-Verbindung_5") mit BSEND gesendeten Daten. Auf einer S7-Verbindung können mehrere zusammengehörende BSEND/BRECV Pärchen existieren, die über ihre RID verbunden sind. Hier ist es der BRECV, der zum BSEND mit der ID 3 gehört.

S7:[<connectionname>]BSEND<length>,<RID>{,{<type>}<address>{,<quantity>}} Beispiel: S7:[S7-Verbindung_2]bsend1024,1,W100,20

© Siemens AG 2019 All rights reserved

Beim Schreiben auf diese Nodeld wird ein Array of Words (unsigned integer 16 Bit) mit 20 Elementen ab der Byte-Offset-Adresse 100 in den Sendepuffer der Länge 1024 geschrieben. Es wird in dem 1024 Byte großen Puffer der Bereich von 100 bis 140 überschrieben. Der gesamte Block wird mit der ID 1 zum Kommunikationspartner gesendet, der einen BRECV mit der ID 1 und einer Mindestlände von 1024 Bytes bereitstellen muss, um die Daten zu empfangen.

Hinweis Für die Verwendung der Blockdienste BSEND/BRCV muss eine beidseitig projektierte Verbindung existieren und die Steuerung muss selbstständig die Bausteine SFB12/13 aufrufen und deren Parameter versorgen.

Zum Thema blockorientierte Dienste beachten Sie auch die Hinweise im SIMATIC NET Handbuch (siehe \7\)

4 Funktionsmechanismen dieser Applikation

Gesamtübersicht

Abbildung 4-1



Nr	Modul	
1.	OPC DA Server (SimaticNET)	Der SimaticNET OPC Server implementiert die notwendige Serverlogik für Groups und Items und die Datenanbindung an die S7 Stationen.
2.	OpcRcw.Da	Schnittstelle, um auf COM-Komponenten zuzugreifen. Kapselt die OPC-Interfaces für den Zugriff aus der .NET Anwendung.
3.	COMDA Client API	Wiederverwendbare, vereinfachte und auf diese Aufgabe zugeschnittene .NET Client API. Sie bietet zum Finden und Verbinden von Servern und zum Beobachten von Werten wieder verwendbare C# Handling Klassen an.
4.	Simple Client	Einfache Bedienoberfläche für die Verwendung der Client API mit den Funktionen Connect, Disconnect, Read, Write und Daten Monitoring.

Programmübersicht

Die folgende Abbildung zeigt die Funktionsblöcke im Simple Client und das Zusammenspiel mit dem OPC COM DA Server.



Nr.	Beschreibung
1	Beim Verbindungsaufbau wird das Server Objekt im lokalen oder entfernten OPC-Server erzeugt.
2	Über den COM Mechanismus CoCreateInstance wird das Server Object erzeugt.
3	Nach dem Verbindungsaufbau legt die Client API legt eine inaktive Gruppe an, die zum Lesen und Schreiben von Items benutzt wird.
4	Beim Anmelden von Variablen zur Beobachtung von Wertänderungen wird auf Clientseite ein Subscription Objekt erzeugt. Logisch gesehen entspricht das einer OPC Group.
5	Über die COM DA Schnittstelle wird im Server ein OPC Group Objekt erzeugt.
6	Die empfangenen DataChanges werden von der ClientAPI an eine Callback- Methode in der Client Applikation weitergeleitet. Die Client Applikation aktualisiert dann die entsprechenden Elemente in der Oberfläche.

4.1 COMDA Client API

Das Klassendiagramm in Abbildung 4-3 zeigt die Klassen der COM DA ClientAPI. Diese Klassen kapseln die Zugriffe auf den OPC-Server in eine vereinfachte und wiederverwendbare .NET API.

Die Klassen sind in der .NET Assembly Siemens.Opc.Da.ClientAPI .dll zusammengefasst. Sie hat Abhängigkeiten zu den .NET RCW Dateien der OPC Foundation Opc.Rcw.Comn.dll und Opc.Rcw.Da.dll.





Klasse Server

Die in der folgenden Tabelle beschriebene Wrapperklasse **Server** kapselt die Funktionalität für den Zugriff auf den OPC-Server. Darüber hinaus vereinfacht sie die Verwendung derjenigen OPC-Interfaces, die von der Client Applikation benötigt werden, mit Ausnahme der Interfaces zum Hinzufügen und Löschen von Items in der Subscription.

Die Klasse ist in der Datei OpcDaServer.cs im Projekt COMDAClientApi implementiert.

Methode	Funktionalität
Connect	Erzeugt eine Instanz des COM Objects IOPCServer. Danach wird auf dem Server eine Gruppe angelegt. Diese Gruppe wird zum Lesen und Schreiben von Items genutzt.
Disconnect	Gibt alle Referenzen auf das COM-Objekt frei und löscht alle Gruppen.
Browse	Bietet ein einheitliches vereinfachtes Interface zum Browsen. Mit dieser Methode kann auf DA2.05 und DA30 Servern gebrowst werden.
Read	Liefert den aktuellen Wert einer Variablen im Server.
Write	Schreibt den Wert einer Variablen auf den Server.
CreateSubscription	Erzeugt eine Subscription. Die Subscription ist der Container um Itemwerte zu beobachten. Zum OPC-Server hin wird eine OPCGroup angelegt. Dieser Funktion wird eine Callback Funktion übergeben. Diese Callback Funktion wird aufgerufen sobald sich Werte für die Items dieser Subscription geändert haben.
DeleteSubscription	Entfernt eine existierende Subscription und löscht die Gruppe auf dem OPC-Server.

Klasse Subscription

Die in der folgenden Tabelle beschriebene Wrapperklasse **Subscription** kapselt die Verwendung einer OPCGroup zum Austausch von Werteänderungen zwischen Server und Client.

Die Klasse ist in der Datei OpcDaSubscription.cs im Projekt COMDAClientAPI implementiert.

Tabell	е	4-4
--------	---	-----

Methode	Funktionalität
AddItem	Erzeugt ein Item in der entsprechenden OPC-Gruppe zur Beobachtung von Werteänderungen und verknüpft dieses mit der Subscription.
Removeltem	Entfernt ein Item aus der Subscription.

4.2 Simple Client COM DA

Der Simple Client stellt ein einfaches Beispiel für die Verwendung der Client API dar. Die wichtigsten Funktionen wie Connect, Disconnect, Read, Write und Monitoring von Daten werden in einer Datei bzw. Klasse mit einem Dialog gezeigt. Der Code für das Beispiel kann im Projekt SimpleClientCOMDA in der Datei SimpleClient.cs gefunden werden.

Oberfläche des einfachen Beispiels

Die Oberfläche wird über Buttons für die einzelnen Funktionen bedient.

Abbildung 4-4

Simple Client OPC COM DA		
Disconnect opcda://localhost/Opc.SimaticNI	1)	OPC DA Server URL
S7 300 - using items on an S7 300 device. Bloc	Read / Write is enabled 9	
S7 1200 - using items on an S7 1200 device. B	ock Read / Write is disable	
	onitored Value	Write Value
Dynar 2 og Types/Double Stop	0,8658283817457 69,1341716182546	Write 35
Static/Same Types/Int	4	4
Block Read	Block Read Result	
S7:[Con 1]BRCV,1,D0,1024	Stop 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10	11 12 13 14 15 16 17 18 19
	1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 27 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44	12B 2C 2D 2E 2F 30 31 32 33 45 46 47 48 49 4A 4B 4C 4D
(7)	4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 50 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78	: 3F 60 61 62 63 64 63 66 7 ; 79 7A 7B 7C 7D 7E 7F 80 81
\smile	92 95 94 95 96 97 98 95 96 06 06 06 06 06 07 90 97 32 96 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB	AC AD AE AF B0 B1 33 35 37 36 SC C2
	D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA D8 DC DD D8 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9	DF E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 A FB FC FD FE FF 00 01 02 03
Block Write		
ItemID Block Write		
S7:[Con1]BSEND,1,D0,1024 1024	Write Length in Byte Write Block 1 Write	[0, 1, 2255] Write Block 2 Write [255, 254,0]

Nr.	Beschreibung
5.	In der Textbox für die Server URL kann die URL des Servers angegeben werden. Die URL setzt sich zusammen aus Rechnername und ProgID des Servers:
	opcda:// <rechnername>/ProgID.</rechnername>
	z. B. für einen lokalen SIMATIC NET Server: opcda://localhost/Opc.SimaticNET.
6.	In den Textboxen für die ItemIDs wird die OPC ItemID angegeben. Diese ID ist im Adressraum des Servers eindeutig.
7.	Über den Button Connect / Disconnect kann die Verbindung zum OPC-Server auf- und abgebaut werden.
8.	Über den Button Monitor wird eine Subscription angelegt und die beiden Items zu einer Gruppe hinzugefügt. Von nun an werden Datenänderungen vom Server an den Client gemeldet und in den beiden Textboxen neben dem Button angezeigt. Fehler werden jeweils an Stelle der Werte angezeigt.
	Der Button ändert nach dem Anlegen der Subscription den Text auf "Stop". Über diesen Button kann die Subscription dann auch wieder gelöscht werden.
9.	Über den Button Read werden die Werte der beiden Variablen mit den angegebenen ItemIDs gelesen und in den Textboxen neben dem Button angezeigt.
	Das Lesen zum Server hin geschieht synchron über die OPC-Methode IOPCSyncIO::Read().
10.	Über den Button Write wird der Wert aus der Textbox neben dem Button auf die durch die ItemID festgelegte Variable geschrieben.
	Für das Schreiben wird der Text aus dem Textfeld als Wert an den Server geschickt.
	Der Server konvertiert die Werte automatisch in den passenden Datentyp des Items.

Nr.	Beschreibung
	Das Schreiben zum Server hin geschieht synchron über die OPC-Methode IOPCSyncIO::Write().
11.	In der Gruppe "Block Read" können Daten empfangen werden, die von der S7 mit dem Blockdienst BSEND aktiv geschickt werden. Dies kann z. B. für das Senden von Ergebnisdaten von der S7 an eine PC-Applikation verwendet werden.
12.	In der Gruppe "Block Write" können Datenblöcke an die S7 geschickt werden, die dort mit dem Blockdienst BRECV empfangen werden. Es können zwei Blöcke mit unterschiedlichen Inhalten geschickt werden. Dies kann z. B. für den Download von Rezeptdaten zur S7 verwendet werden.
13.	Die Blockdienste sind nur bei der S7-300 verfügbar. Beim Umschalten auf S7-1200 wird das Lesen und Schreiben von Blocks in der Oberfläche deaktiviert.

Funktionen des einfachen Beispiels

Die Funktionen finden sich in der Klasse SimpleClientDA in der Datei SimpleClient.cs. In den Funktionen ist ein einfaches Fehlerhandling implementiert. Tritt bei OPC-Aufrufen eine Exception auf, wird ein Dialog mit der Fehlermeldung aufgeblendet. Tritt bei den variablenbezogenen Aufrufen ein Fehler bei einer oder mehreren Variablen auf, wird der Fehler in den zugehörigen Textboxen ausgegeben.

Funktion	Beschreibung
btnConnect_Click	In dieser Funktion wird die Verbindung zum OPC-Server über die Funktion Server::Connect() der Client API aufgebaut. Dabei wird der URL-String aus der entsprechenden Textbox übergeben.
btnDisconnect_Click	In dieser Funktion wird die Verbindung zum OPC-Server über die Funktion Server::Disconnect() der Client API abgebaut.
btnRead_Click	In dieser Funktion werden für die beiden Items die Werte über die Funktion Server::Read() gelesen. Das Ergebnis wird in die zugehörige Textbox geschrieben. Tritt beim Lesen ein Fehler auf oder ist die Quality schlecht (d. h. es kann vom Server kein Wert geliefert werden) dann wird der Fehlercode in die Textbox geschrieben.
btnMonitor_Click	Zuerst wird mit Server::CreateSubscription() eine Subscription angelegt. Danach werden die beiden Items mit Subscription::AddItem() angelegt. Die ClientHandles sind in diesem Beispiel fest eingestellt da nur 3 Items benutzt werden. Das ClientHandle wird vom Server bei einer Meldung von Werteänderungen mit an den Client übertragen. Dadurch kann der Client die Werte eindeutig den Items zuordnen. Ist bereits eine Subscription angelegt, wird diese mit Server::DeleteSubscription() gelöscht und der Client empfängt keine Werteänderungen mehr.
OnDataChange	Die Funktion wird bei Server::CreateSubscription() als Callback Funktion angegeben. In der Funktion wird zuerst geprüft, ob der Aufruf im Haupt-Thread des Dialogs ankommt. Wenn nicht, wird der Aufruf über BeginInvoke an den Haupt-Thread des Dialogs übergeben. Ansonsten ist ein Zugriff auf den Dialog nicht möglich. Danach werden die ClientHandles geprüft und anhand der Handles das entsprechende Textfeld aktualisiert. Handelt es sich um den Wert einer normalen Variablen, wird der Wert einfach als Text ausgegeben. Handelt es sich dagegen um den Wert einer Blockvariablen, wird das Bytearray extrahiert und als Folge von HEX-Werten für die einzelnen Elemente des Bytearrays ausgegeben.
btnWrite_Click	Die Funktion ruft Server::Write() auf und schreibt den aktuellen Text als

Funktion	Beschreibung
	Wert auf das Item.
btnMonitorBlock_Click	Zuerst wird mit Server::CreateSubscription() eine Subscription angelegt. Danach wird das Item "BlockWrite" mit Subscription::AddItem() angelegt. Ist bereits eine Subscription angelegt, wird diese mit Server::DeleteSubscription() gelöscht und der Client empfängt keine
	Werteänderungen mehr für das "BlockWrite" Item.
btnWriteBlock1_Click	Die Funktion schreibt generierte Werte auf das "BlockWrite" Item. Die Simulation legt ein Bytearray mit der in der Oberfläche angegeben Länge an und befüllt die Werte mit einem Variablenwert, der bei 0 beginnt und nach jeder Zuweisung inkrementiert wird.
btnWriteBlock2_Click	Die Funktion ist identisch zu btnWriteBlock1_Click; allerdings beginnt der hier zugewiesene Variablenwert mit 255 und wird nach jeder Zuweisung dekrementiert.

Hinweis Weitere Details sind im Source Code dieser Applikation als Kommentar enthalten.

4.3 S7 Programm

Übersicht

Das S7 Programm teilt sich im Wesentlichen in zwei Teile. Zunächst werden die dynamischen Daten für die Variablendienste simuliert, anschließend werden die Sendedaten simuliert und im FB 100 werden die Blockdienste aufgerufen.

Hinweis Die Blockdienste gibt es nur im der S7-300, die S7-1200 kommuniziert ausschließlich über S7 Variablendienste.

Abbildung 4-5



Simulation dynamische Daten

Die folgende Tabelle gibt eine kurze Übersicht über die Programmteile und ihre Funktion zur Datensimulation. Auf Details wird hier bewusst verzichtet.

Tabelle 4-7		
Baustein	Anmerkung	
OB1	Cyclic Main hier wird zunächst ein variabler Timer gesetzt, mit dessen Intervall die weiteren Programmfunktionen aufgerufen werden. Die Datenänderungsgeschwindigkeit kann über DB10 Byte 0 gesetzt werden.	
FC10	ChangeDateAndTime inkrementiert einen Datumswert sowie Uhrzeit im DB51.	
FC11	ChangeSimpleTypes inkrementiert die Daten des DB51. 8Bit Typen werden mit +1, 16Bit Typen mit +100 und 32Bit Typen mit +1000 inkrementiert.	
FC13	ChangeString inkrementiert einen String der Länge 10 im DB51.	
DB10	SimulationConfiguration enthält globale Variablen zur Konfiguration der Datensimulation.	
DB50	StaticDataTypes enthält einfache Datentypen, denen symbolische Namen gegeben wurden. Die Werte sind mit maximalem Wertebereichsendwert vorinitialisiert.	
DB51	DynamicDataTypes enthält einfache Datentypen, denen symbolische Namen gegeben wurden. Die Werte werden mit den Funktionen FC10 bis FC12 entsprechend ihrer Wertebereiche inkrementiert.	
SFC21	FILL Hilfsfunktion, um Datenbereiche mit Werten zu befüllen, Speicherinitialisierung.	

Blockorientierte Daten

Die S7-Kommunikation bei der S7-300 wird über FBs (ladbare Funktionsbausteine) realisiert und nicht anhand von SFBs (integrierten Systemfunktionsbausteinen) wie bei der S7-400. Wenn Sie im S7-Programm der S7-300 jedoch einen SFB statt FB aufrufen, liefert der Baustein einen "ERROR" und gibt den "STATUS = 27" aus. Dieser Status weist darauf hin, dass der Funktionsbaustein für die S7-Kommunikation auf der S7-300 nicht vorhanden ist. Die Kommunikations-FBs für die S7-300 befinden sich in der Bibliothek "SIMATIC_NET_CP > CP300 > Bausteine".

Die folgende Tabelle gibt eine kurze Übersicht über die Programmteile und ihre Funktion bezüglich BSEND/BRCV. Auf Einzelheiten wird hier bewusst verzichtet. Tabelle 4-8

Baustein	Anmerkung
OB1	Cyclic Main Aufruf des Sendebausteins (SFB12) für BSEND und des Empfangsbausteins (SFB13) für BRCV über den Funktionsblock 100. Für S7-300 werden FB12 und FB13 aus der Bibliothek verwendet da die S7-300 über ladbare FBs und nicht über
	Systemfunktionen kommuniziert.
FC14	ChangeSendData inkrementiert ein Byte, welches dann auf den gesamten Sendepuffer kopiert wird.
FB100 + DB100 (Instanz- DB)	InvokeBSENDandBRCV ruft die echten Kommunikationsbausteine auf und versorg deren Parameter.
DB112	SendData Datenbaustein mit 4096 Bytes Länge, der dem Sendebaustein übergeben wird.
DB113	ReceiveData Datenbaustein mit 4096 Bytes Länge, der dem Empfangsbaustein übergeben wird.
SFB12 + DB12 (Instanz- DB)	BSEND Der Sendebaustein übergibt den Sendepuffer an den Kommunikationsprozessor (CP), der ihn dann entsprechend RID und VerbindungsID an den Kommunikationspartner sendet.
FB12	BSEND (nur S7-300)
SFB13 + DB13 (Instanz- DB)	BRCV Der Empfangsbaustein holt das zuletzt empfangene Datenpaket entsprechend RID und VerbindungsID vom Kommunikationsprozessor (CP) ab und legt es in den Empfangspuffer.
FB13	BRCV (nur S7-300)
SFC21	FILL Hilfsfunktion um Datenbereiche mit Werten zu befüllen, Speicherinitialisierung.

Die Programmlogik sendet bzw. empfängt Datenblöcke und versorgt die entsprechenden Parameter über den FB100. Wenn größere Datenpakete gesendet werden, dann ist ein mehrfacher Aufruf des BSEND bzw. BRCV Bausteins erforderlich. Dieser Mehrfachaufruf wird vom FB100 durchgeführt.

Hinweis Werden Sende- und Empfangsbaustein zyklisch aufgerufen, kann zum einen eine hohe Kommunikationslast entstehen und zum anderen besteht die Gefahr, dass die Datenpuffer überschrieben werden, bevor sie von der Gegenseite verarbeitet werden konnten. Die Programmlogik sollte deshalb eine Flusskontrolle beinhalten, um die Datenkonsistenz sicherzustellen. Hierzu sind der Parameter DONE (fertig) und NDR (neue Daten empfangen) zu verwenden.

Um Daten zwischen zwei S7-300 Stationen über eine im TIA-Protal projektierte S7-Verbindung austauschen zu können, müssen im S7-Programm Kommunikationsfunktionen aufgerufen werden. Der Baustein FB12 "BSEND" dient zum Senden von Daten und der Baustein FB13 "BRCV" zum Empfangen von Daten.

Hierbei muss die S7-Verbindung beidseitig projektiert werden, da die S7-Kommunikation mittels FB12 "BSEND" und FB13 "BRCV" auf dem Client-Client Prinzip basiert.

Hinweis Wenn die S7-Verbindung über die integrierte IE-Schnittstelle der S7-300 Steuerungen der CPU31x-2PN/DP oder der CPU319-3PN/DP projektiert ist, dann müssen der FB12 "BSEND" und FB13 "BRCV" aus der Bibliothek "Standard Library -> Communication Blocks -> Blocks" mit der Familie="CPU_300" verwendet werden. Diese FBs sind für die S7-Kommunikation über die integrierte IE-Schnittstelle der CPU sowie für die S7-Kommunikation über die S7-300 IE-CPs verwendbar.

5 Konfiguration und Projektierung des OPC-Servers

Einleitung

In den folgenden Abschnitten zeigen wir Ihnen die Schritte, um die PC-Station und den OPC-Server in STEP7 V11 zu projektieren. Sie müssen dieses Kapitel nur lesen, wenn Sie an den Details interessiert sind. Die Projektierung ist im mitgelieferten STEP7 Projekt bereits vollständig erfolgt.

5.1 Projektierung des OPC-Servers in STEP7 V15.1

Führen Sie folgende Schritte durch, um z. B. zu einem vorhandenen STEP 7 V15.1-Projekt(oder höher) mit bereits projektierter SIMATIC-Station einen OPC-Server hinzuzufügen.

Tabelle 5-1





1.1.	11	- 1 - D	
Inna	itsverz	eicn	nıs



Nr.	Aktion	Anmerkung		
6	Aktivieren Sie die symbolische Darstellung der OPC Variablen Bei der Benutzung von Symbolen können Sie konfigurieren ob sie "Alle" [All] oder "nur bestimmte" Symbole [Configured] verwenden wollen. Oftmals möchte man nicht alle Symbole über den OPC-Server verfügbar machen (z. B. interne Variablen oder Inhalte von Instanzdatenbausteinen).			
	> Dptions Iools Window Help D:\step7_projects\TiaPortal\OPCSample\OPCSample\OPCSample Iy Integrated Automation > X > ± C* ± Im Im Im Im PORTAL OPCSample > OPC-Server [SIMATIC PC station] Im Im			
	🔐 🖸 🖂			
	PC station	Catalog Vare catalog Vare catalog Vare catalog Search> Imm Filter SIMATIC Controller Ap SIMATIC Controller Ap SIMATIC Controller Ap SIMATIC Controller Ap December 2 Superior Communications mo PROFINET/Ethernet POFINET/Ethernet Communications mo Controller Ap Controler Ap Controller Ap Controller Ap Controle		
	< III	WinAC communics		
	OPC-Server [OPC Server]	Device data Properties Linfo Diagnostics Configuring risible during runtime		
	r B OPC-Server	Project OPCSample opened.		



Im nächsten Kapitel wird dies gezeigt.

5.2 **Projektierung der S7 Verbindungen**

Die PC-Station und der OPC-Server benötigen projektierte S7 Verbindungen, um die symbolische Darstellung der OPC-Items verwenden zu können.

Tabelle 5-2



Inhaltsver	zeich	nis
in in lance v cr	20101	1110





Inhaltsve	rzeichnis
IIIIIaitsve	

Nr.	Aktion	Anmerkung
6	Konfigurieren Sie die Verb	indung, indem Sie die Eigenschaften der Verbindung editieren.
	Properties	
	S7 connection: 300	🔍 Properties 🚺 Info 🚺 💆 Diagnostics
	General General Local ID Special connection properties Address details OPC Alarms Diagnostics alarms	General Connection Offline status: So Name: 300 Connection path Local Partner Office Server Sr-300 Interface: IE General, PROFINET inte CP 343-1, PROFINET inte Interface type: Ethernet/IP Ethernet/I
7	Die Verbindung zur CPU 3	15-2 PN/DP wurde hier beidseitig projektiert
	300 [S7 connection]	
	General IO tags	System constants Texts
	General Local ID	Special connection properties
	Special connection prope Address details	Local end point
	OPC	One-way
	Alarms Diagnostics alarms	Active connection establishment Send operating mode messages

Nr.	Aktion	Anmerkung		
9	Der OPC-Server sollte die Weiterhin sollte der OPC- erkannt wird.	e Verbindung immer aktiv aufbauen und permanent aufrecht erhalten. Server unverzüglich reagieren, falls eine Verbindungsunterbrechung		
	Properties S7 connection: 300			
	General			
	General Local ID Special connection properties Address details OPC Alarms Diagnostics alarms Adarms Adarms Adarms Adarms Adarms			
		Default priority for alarms: 500 Receive block-related alarms (interrupts as conditional events) Receive block-related alarms (interrupts as simple events) Receive diagnostics alarms		
		Other parameters Optimize write access Optimize read access Automatic reset of 37 password for block access Immediate response when interrupted connection is detected		
		Disconnect automatically after 0 s Timeout during connection establishment 15000 ms Job timeout 15000 ms Maximum number of parallel network jobs 2 PDU size 240 Byte		
		•		
10	Führen sie die selben Sch	aritte analog für die SZ-Verbindung zur SZ-1200 durch		
10	Consistente Cie Ibr Desistet			
11	Speichern Sie inr Projekt.			

5.3 Kontrollieren der Einstellungen

Die Einstellungen können mit der Konfigurations-Konsole "PC-Station einstellen" [Communication Settings] kontrolliert werden.

Dabei sollte vor allem auf die folgenden Einstellungen geachtet werden:

Tabelle 5-3



Nr.	Aktion	Anmerkung		
4	Kontrollieren Sie eingestellte	en Protokolle		
	Hinweis: Für diese Applikation genügt die Aktivierung des S7-Protokolls.			
	Siemens Communication Settings			
	File Language Help			
	GPC settings	PC protocol selection		
	🛃 👌 Quit OPC server			
	Select OPC protocol	The OPC Server can support various protocols at the same time. Here, you select the protocols that will be supported.		
	A Security	Name: OPC UA Details		
	OPC UA certificates	DP AU		
	Image Settings Image Se	DP master class 2		
	▼ Modules			
	▶ 🛄 Intel(R) 82566DM-2 Gig 🗹	SR AU		
	General	SINMP		
	Address	AU AU		
	TCP parameters	Operate XML OPC server along with OPC server. This means that fast in-process OPC servers		
	U COMES/	are not possible!		
	🗓 S7 test	Apply Cancel		
	😲 SR test			
	Access points			
5	Kontrollieren Sie, ob die Symbolik geladen wurde.			
	Siemens Communication Settings	- 🗆 ×		
	<u>File L</u> anguage <u>H</u> elp			
	✓ SIMATIC NET configuration ✓ S ✓ S ✓ S ✓ OPC settings	elected symbol files		
	📕 🕘 Quit OPC server	1 🕅 🤇		
	Select OPC protocol	If you want to use symbolic names for the variables, you need to create a symbol file. Please specifiy an		
	Security	existing symbol me or create a new symbol me with the symbol curtor.		
	🔐 OPC UA certificates 🛛 🔵	Active symbol files:		
	Trace settings	C:\ProgramData\Siemens\SIMATIC.NET\opc2\binS7\sym		
	Modules			
	▶ 🛄 Intel(R) 82566DM-2 Gig 🗹	Additional settings for SIMOTION:		
	Intel(R) Gigabit CT De			
	Address	The symbol files are in use. The current configuration cannot be modified. Quit the OPC server		
	TCP parameters	If you use the XML OPC Server, also quit the IIS.		
	COML S7	Apply Cancel		
	S7 test			
	💟 SR test 🔷			
	Access points			
6	Sollte eine der Einstellunger	n nicht den gezeigten Bildern entsprechen, so führen die		
	vorhergehenden Projektieru	ngen erneut aus.		
	Schließen Sie den Konfigura	ationsdialog.		

6 Installation und Inbetriebnahme

6.1 Installation der Hard- und Software

In diesem Kapitel beschreiben wir, welche Hardware- und Softwarekomponenten Sie installieren müssen, um das Beispiel mit den vorhandenen Projektierungsdaten in Betrieb zu nehmen. Beachten Sie zudem die Handbücher sowie Lieferinformationen, die mit den entsprechenden Produkten ausgeliefert werden.

Installation der Hardware

Die Hardware-Komponenten entnehmen Sie bitte dem Kap. 2.3. Gehen Sie für den Hardwareaufbau gemäß folgender Tabelle vor:

ACHTUNG Schalten Sie die Spannungsversorgung erst nach dem letzten Schritt zu.

Tabelle 6-1

Nr.	Fokus	Aktion
1	Steuerung – S7-300 Station	Bauen Sie die Station analog zur Abbildung in Kap. 2 auf.
2	Steuerung – S7-1200 Station	Bauen Sie die Station analog zur Abbildung in Kap. 2 auf.
3	PG/PC Station	Bauen Sie die Station analog zur Abbildung in Kap.2 auf.
4	Industrial Ethernet	Verbinden Sie die Steuerung mit dem PG analog zur Abbildung in Kap. 2.

Hinweis Anstatt eines Hubs oder Switch können Sie auch ein CrossCable für eine direkte Verbindung verwenden.

Beachten Sie generell die Aufbaurichtlinien für SIMATIC S7.

Installation der Standard Software

Auf dem PG/PC muss STEP7 V15.1 und SIMATIC NET installiert sein. Auf aktuellen SIMATIC PGs ist STEP7 V15.1 bereits vorinstalliert.

Auf die Beschreibung der Installation von STEP7 und SIMATIC NET wird an dieser Stelle verzichtet. Die Installation findet in gewohnter Windows-Umgebung statt und ist selbsterklärend bzw. in den entsprechenden Handbüchern beschrieben.

Adressübersicht der beteiligten Baugruppen

Falls Sie das Projekt an einem vorhandenen Industrial Ethernet betreiben wollen, müssen Sie folgende Adressvergabe beachten:

Fokus	Baugruppe	IP-Adresse
PG/PC	NDIS-Netzwerkkarte	192.168.172.1
Steuerung	CP 343-1	192.168.172.2
Steuerung	CPU 1214 C	192.168.172.4

Hinweise

- Beachten Sie die richtige Subnetz-Maske 255.255.255.0.
- Alternativ ist es auch möglich, im STEP7-Projekt die zugewiesenen IP-Adressen zu ändern.

Einstellen der IP Adresse

Die Ethernet-Netzwerkkarte muss in den projektierten Betrieb geschaltet werden. Hierzu muss die PC-Station konfiguriert werden.

Hinweis Stellen Sie sicher, dass die Netzwerkkarte die feste IP-Adresse 192.168.172.1 hat (diese können Sie über die Netzwerkeinstellungen und den TCP/IP-Eigenschaften einstellen), wenn Sie das mitgelieferte Projekt verwenden wollen.

6.2 Laden der der PC Station über STEP 7 V1x

Installation des STEP7 V1x Projekts via TIA Portal

Die PC-Station kann direkt aus dem TIA-Portal geladen werden. Alternativ hierzu kann eine PC-Station auch über den Komponenten Konfigurator und die XDB-Datei konfiguriert werden (Siehe Kap. 6.3).

Die Beschreibung in Tabelle **Fehler! Verweisquelle konnte nicht gefunden w** erden. ist mit TIA V11 erstellt worden. Das Vorgehen für TIA V15.1 ist analog

Tahal	ماا	6-3
raper	ne	0-3

Nr.	Aktion	Anmerkung
1	Entpacken Sie das TIA-Projekt: STEP7_TIA15.1.zip	Entpacken Sie das Projekt in einem Pfad in dem Sie Lese- und Schreibrechte besitzen.
2	Öffnen Sie das TIA-Portal und navigieren Sie zum Projekt mittels der Browserfunktion	Financial Stat
3	Bestätigen Sie mit Öffnen.	VOPCSample/OPCSample.ap11 Ware Look /r: DPCSample Det modified Type AdditionalFiles 6/8/2012 6:21 PM File folder File folder Desktop System 6/8/2012 6:21 PM Desktop System 6/8/2012 6:21 PM UserFiles 6/8/2012 6:21 PM File folder UserFiles 6/8/2012 6:21 PM File folder Dibraries OPCSample.ap11 6/11/2012 6:46 PM Siemens T Network Tim File folder File folder File game: OPCSample.ap11 OPCS Open Cancel
4	Nach dem Öffnen wechseln Sie in die Projektansicht	Contract-of-Changed Contraction Dest: 14 / Yes: plot Diff: 0 / Yes Total A / Y / Y / Y / Y / Y / Y / Y / Y / Y /

Nr.	Aktion	Anmerkung
5	Laden Sie die PC-Station Alternative: Sie können die PC-Station auch über den Import der mitgelieferten XDB-Datei konfigurieren (Siehe Kap 6.3).	International Control of Con

Ändern der IP Adresse der PC Station in STEP7 V1x

Hinweis Führen Sie diese Schritte nur aus, wenn Sie die IP-Adresse Ihrer PC-Station ändern möchten.

Tabelle 6-4





6.3 Importieren der XDB-Datei in den Komponenten Konfigurator

Einleitung

Eine PC-Station kann alternativ zum Laden aus dem TIA-Portal (Siehe Kap **Fehler! V** erweisquelle konnte nicht gefunden werden.) auch über den Komponenten Konfigurator und die XDB-Datei konfiguriert werden. Die XDB-Datei ist im mitgelieferten TIA-Projekt bereits vorhanden.

Einstellen der IP Adresse

Die Ethernet-Netzwerkkarte muss in den projektierten Betrieb geschaltet werden. Hierzu muss die PC-Station konfiguriert werden.

Hinweis Stellen Sie sicher, dass die Netzwerkkarte die feste IP-Adresse 192.168.172.1 hat (diese können Sie über die Netzwerkeinstellungen und den TCP/IP-Eigenschaften einstellen), wenn Sie das mitgelieferte Projekt verwenden wollen.

Tabelle 6-5



Inhaltsverzeichnis

Nr.	Aktion	Anmerkung	g		
3	Navigieren Sie zum Projektordner Ihres STEP7	Import XDB file	Search OPCSemple		
	V15.1-Projekts. Selektieren Sie	Organize New folder			
	die XDB-Datei.	Superior Name	Date modified Type		
	Betatigen Sie den Dialog.	Favorres Favorres E Desktop Downloads Downloads M	6/8/2012 6:21 PM File folder 6/8/2012 6:21 PM File folder		
		Libraries	6/8/2012 6:21 PM File folder 6/8/2012 6:21 PM File folder 6/8/2012 6:21 PM File folder 6/8/2012 6:21 PM File folder		
		Music Pictures Videos OPC-Server.xdb	6/11/2012 6:43 PM XDB File		
		🜏 Homegroup			
		👰 Computer	,		
		File <u>n</u> ame: OPC-Server.xdb	▼ *xdb ▼		
			Open Cancel		
4	Der Import-Assistent bestätigt nun. dass der Import möglich ist.	Configuration for XDB Import			
		Index Name Type Status	Error		
	Betätigen Sie mit OK.	1 IF IE General IE General	Configured component was rep Configured component was rep		
	Hinweis:	4	E		
	Falls Komponenten in einer	6			
	wurden, werden diese durch	7			
	existierende kompatible	9			
	Versionen getauscht	10			
		12			
		13			
		14			
		15			
The XDB import is possible. Refer to the list above for the configuration					
		Configured component was replaced by an existing compatible component!			
		ОК	Cancel Help		

	-		
Inho	Ito' or	ninh	nin
Inna	IISverz	есп	I IIS
	1001012		

Nr.	Aktion	Anmerkung	
5	5 Nach dem erfolgreichen Import der XDB-Datei, ist Ihre PC- Station im Zustand ONLINE.	Station Configuration Editor - [ONLINE] Components Diagnostics Components Diagnostics	
		Station: OPC-Server Mode: RUN_P	
		Index Name Type Ring Status Run/Stop Conn	
		Index Name Type Hing Status Hunristop Com 1 IE General IE General Image: Com Imag	E
	16 17 Add Edit Delete Ring DN Station Name Import Station Disable Station	tion	

6.4 Installation des OPC-Clients auf dem PC/PG

Die Applikations-Software wird mit einem Setup-Programm geliefert.

ACHTUNG Falls Sie ein älteres Betriebssystem als Windows 7 SP1 nutzen und dort nicht die SimaticNET PC Software V8.x installiert wurde, so müssen Sie zuerst das .NET-Framework 3.5 +SP1 installieren.

Informationen hierzu erhalten Sie auf den Microsoft Internet Seiten (siehe \5\)

Gehen Sie zur Installation der Bedienoberfläche folgendermaßen vor:

Tabelle 6-6

Nr.	Aktion	Anmerkung
1.	Entpacken Sie die Datei 21043779_OPCClient_RCW_CODE.zip	Diese zip-Datei enthält sowohl das STEP7 V11 und das STEP7 V13 Projekt, als auch den OPC-Client mit C# Source- Code.
2.	Entpacken Sie die Datei Csharp_OPCClient_RCW_CODE.zip	
3.	Im Verzeichnis \OpcClientDA_V2\bin finden Sie die Datei: DAClient.exe	Die EXE kann nur ausgeführt werden wenn die dazugehörigen Assemblies im selben Verzeichnis liegen

Enthaltene Dateien

Die Archivdatei enthält die MS Visual Studio Solution Datei und den Source Code sowie vorkompilierte Binärdateien für x86 Systeme. Im Unterordner befindet sich die ausführbare Datei (EXE) sowie die benötigten Assemblies.

Verzeichnis: \OpcClientDA_V2\bin

Tabelle 6-7

Datei	Gehört zu
DAClient.exe	Hauptanwendung
OpcCRcw.Comn.dll	OPC-RC-Wrapper
OpcRcw.Da.dll	OPC-RC-Wrapper
ClientAPI.dll	Wiederverwendbare OPC DA Funktionen

6.5 Laden der Simulation auf die S7-Stationen

Laden des TIA-Projekts

Gehen Sie dazu wie folgt vor:

Tabelle 6-8

Nr.	Aktion	Anmerkung
1	Entpacken Sie das TIA- Projekt: STEP7_TIA15.1.zip	Entpacken Sie das Projekt in einem Pfad in dem Sie Lese- und Schreibrechte besitzen.
2	Öffnen Sie das TIA- Portal und navigieren Sie zum Projekt mittels der Browserfunktion	Sterens Totally Integrated Automation PORTAL Start Image: Compare addition and order project Device & A Image: Compare addition and order project Device & A Image: Compare addition and order project Device & A Image: Compare addition and order project Device & Compare addition and order project Image: Compare addition and order project Device & Compare addition and order project Image: Compare addition and order project Outline & Compare addition addition addition additional document additional additional document additional additional additional additional additional additional additional additional additional add
3	Kompilieren und Laden Sie die S7-300 Station	Stemense - OrkSample Control Project Edit Very Control Very Control Project Edit Project Control Project Control Project Control Project Control<
4	Kompilieren und Laden Sie die S7-1200 Station.	Wareness - Ork Sample

7 Bedienung der Applikation

7.1 Übersicht

Der erste Teil zur Bedienung der Applikation zeigt wie Daten von S7 Variablen über direkte bzw. symbolische Adressierung beobachtet, gelesen und geschrieben werden.

Tabelle 7-1

Nr.	Aktion	Anmerkung		
1 Starten des		Nach dem Starten der Anwendung erscheint die Benutzeroberfläche.		
	Simple	Simple Client OPC COM DA		
	Olicitia	Connect opcda://localhost/Opc.SmaticNET OPC DA Server URL		
		S 7 300 - using items on an S7 1200 device. Block Read / Write is disabled		
		ItemID Monitored Value Read Value Write Value		
		Dynamic/Analog Types/Double Monitor Read Write		
		Block Read ItemID Block Read Block Read Result		
		S7;{Con1]BRCV.1.00.1024 Monitor Block		
		Block Write temID Block Write Write 57:[Con1]BSEND,1.00.1024 1024 Write Length in Byte Write Block 1 Write [0, 1, 2255] Write Block 2 Write [255, 254,0]		
2 Verbindung zum Server aufbauen Die Server URL muss manuell eingegeben werden. Über den Connect Button kann die Verbindung zum Server		Die Server URL muss manuell eingegeben werden. Über den Connect Button kann die Verbindung zum Server aufgebaut werden.		
		Simple Client OPC COM DA		
		Connect opcda://localhost/Opc.SimaticNET OPC DA Server URL		
3	3Typ der Steuerung auswählen.Die Verwendung von Blockdiensten ist in dieser Applikation nur mit der Familie S7-300 möglich. Bei Wahl der S7-1200 werden die Oberflächenelemente für Blockdienste deaktiviert.			
		🖳 Simple Client OPC COM DA		
		Disconnect opcda://localhost/Opc.SimaticNET		
		S7 300 - using items on an S7 300 device. Block Read / Write is enabled		
		S7 1200 - using items on an S7 1200 device. Block Read / Write is disabled		

Nr.	Aktion	Anmerkung		
4	Monitoring von Daten	Für das Beobachten von Datenänderungen, Lesen und Schreiben können zwei Variablen angegeben werden. Als ItemID wird entweder der symbolische Name im OPC-Server angegeben oder direkt adressiert (z. B. S7:[con1]MB100). Über den Button Monitor kann die Beobachtung eingeschaltet werden. Die gemeldeten Datenänderungen werden in den Textfeldern neben dem Button angezeigt. Ist die Beobachtung aktiv, ändert der Button den Text auf Stop. Image: Simple Client OPC COM DA Image: Disconnect opcda://localhost/Opc.SimaticNET Image: Simple Client opc and stop in the stop in		
5	Lesen und Schreiben	Für das Lesen und Schreiben werden die gleichen ItemIDs verwendet wie für das Beobachten. Durch Klicken der Schaltfläche "Read" werden beide Variablen gelesen. Über die beiden "Write"-Schaltflächen können die Variablen einzeln geschrieben werden. Read Value Read 69,1341716182545 2147483647 Write		

7.2 Verwenden der Blockdienste

Verwendung der Blockdienste

Als anschauliches, einfaches Beispiel wird hier ein typischer Anwendungsfall für blockorientierte Dienste gezeigt: Das Senden von Rezeptdaten an die S7 bzw. das Empfangen von Ergebnisdaten von der S7 unter Verwendung der Blockdienste BSEND und BRCV.

Der Server entscheidet anhand der Syntax der ItemIDs über welchen Dienst mit der S7 kommuniziert wird z. B. S7:[S7Verbindung1]BRCV,1,D0,1024].

Für die Verwendung der Blockdienste muss also eine entsprechende ItemID in den Feldern "ItemID Block Read" bzw. "ItemID Block Write" eingetragen werden.

Tabelle 7-2

Nr.	Aktion	Anmerkung
1. S (n	Starten des Simple Clients und verbinden mit dem S7 Server	Simple Client OPC COM DA COM DA Comparing A static NET OPC DA Server UPL S7 300 - using items on an S7 300 device. Block. Read / Write is enabled S7 1200 - using items on an S7 1020 device. Block. Read / Write is deabled temID Montored Value Montored Value Value Montored Valu
		Biock Wite Book Wite Book Wite String Service String Service Wite Block Wite String Service String Service Wite Block Wite String Service String Service Wite Block Wite String Service Wite Block Wite
2.	Empfangen von der S7	Zum Empfangen von der S7 muss "Monitor Block" aktiviert werden. Sobald das Monitoring aktiv ist ändert der Button den Text auf "Stop". Im Textfeld "Block Read Result" werden die Daten des gelesenen Blocks als HEX-Code angezeigt. Block Read ItemID Block Read Stop 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 82 83 84 88 68 78 88 98 A8 B8 C 8D 8E 8F 9 CD 9F FA0 A1 A2 A3 A4 A5 A6 A7 A8/ B6 6F 7B 8B 9B AB BB CD DE BF C0 C1 C2 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB C
3.	Senden der Daten an die S7	Wählen Sie das "Rezept", in dem Sie entweder "Write Block 1" oder "Write Block 2" drücken. Der erste Button schreibt in den Datenblock, indem ein Wert für jeden Arrayeintrag ab 0 inkrementiert wird. Beim zweiten werden die Arrayeinträge ab 255 dekrementiert. Block Write teml D Block Write S7:[Con1]BSEND.1.D0.1024 Write Block 1

8 Glossar

COM / DCOM

COM (Component Object Modell): Software-Modell zur Kommunikation zwischen Komponenten, das auf einer einheitlichen Schnittstelle aufsetzt. DCOM: Software-Modell zur Kommunikation über Rechnergrenzen hinweg, welches auf COM aufsetzt.

Event-Handler

Ein Event-Handler bearbeitet aufgetretene Ereignisse bzw. Windows-Nachrichten.

Exception

Unter einer Exception versteht man eine Ausnahmesituation. Diese kann entweder vom Betriebssystem (z. B. Division durch Null) oder vom Anwenderprogramm generiert werden.

Exception-Handler

Ein Exception-Handler bearbeitet aufgetretene Ausnahmesituationen. Dies ist in der Regel ein gesichertes Fehlverhalten und/oder eine Nachricht an den Nutzer.

HRESULT

Rückgabe-Datentyp von COM-Objekten.

IDL

Interface Definition Language: Eine von Microsoft standardisierte Sprache zur Definition von Funktions- und Parameter-Schnittstellen.

Polling

Englischer Begriff, mit dem das (meist zyklische) Abfragen bestimmter Werte oder Zustände gemeint ist.

Sink-Interface

Mit Hilfe des Sink-Interfaces können Nachrichten zwischen Komponenten gesendet werden. Das Sink-Interface setzt auf COM-Mechanismen auf.

Thread

Durch "Threads" lassen sich innerhalb einer Anwendung bzw. eines Prozesses mehrere Code-Fragmente quasi parallel, also gleichzeitig, ausführen.

Verwendet eine Applikation mehrere Threads, so besitzt die Applikation die Eigenschaft "multi-threaded".

Hat eine Applikation ausschließlich einen Thread, so heißt sie "single-threaded". Bei diesen Applikationen werden somit alle Code-Fragmente immer sequentiell abgearbeitet.

Windows-Nachricht

In den gängigen Microsoft Windows Betriebssystemen werden zur Mitteilung von Ereignissen, z. B. das Paint-Ereignis, Nachrichten ausgetauscht.

Wrapper

Als "Wrapper" wird in der Regel ein Klassenverbund bezeichnet, der andere Klassenverbände zur Datenkonvertierung oder leichteren Verwendung kapselt. Er kann somit als "Hülle" gesehen werden, der die "gewrappten" Klassen umschließt und nach außen verbirgt.

9 Literaturhinweise

9.1 Literaturangaben

Diese Liste ist keinesfalls vollständig und spiegelt nur eine Auswahl an geeigneter Literatur wieder.

Themengebiet	Titel
STEP7 SIMATIC S7-300/400	Automatisieren mit STEP7 in AWL und SCL Autor: Hans Berger Publicis MCD Verlag ISBN: 978-3-89578-397-5
STEP7 SIMATIC S7-300/400	Automatisieren mit STEP7 in KOP und FUP Autor: Hans Berger Publicis MCD Verlag ISBN: 978-3-89578-296-1
STEP7 SIMATIC S7-300	Automatisieren mit SIMATIC S7-300 im TIA-Portal Autor: Hans Berger Publicis MCD Verlag ISBN: 978-3-89578-357-9
STEP7 SIMATIC S7-400	Automatisieren mit SIMATIC S7-400 im TIA-Portal Autor: Hans Berger Publicis MCD Verlag ISBN: 978-3-89578-372-2
STEP7 SIMATIC S7-1200	Automatisieren mit SIMATIC S7-1200 Autor: Hans Berger Publicis MCD Verlag ISBN: 978-3-89578-355-5
	ThemengebietSTEP7 SIMATIC S7-300/400STEP7 SIMATIC S7-300/400STEP7 SIMATIC S7-300STEP7 SIMATIC S7-400STEP7 SIMATIC S7-1200

Tabelle 9-1

9.2 Internet-Link-Angaben

Diese Liste ist keinesfalls vollständig und spiegelt nur eine Auswahl an geeigneten Informationen wieder.

Tabelle 9-2

	Themengebiet	Titel
\1\	OPC	OPC DA 2.05 Specification bei <u>http://www.opcfoundation.org/</u>
\2\	.NET	Inside C#, Tom Archer
		 .NET Crashkurs, Clemens Vasters, Oellers, Javidi, Jung, Freiberger, DePetrillo
		 Microsoft .NET Framework Programmierung, Jeffrey Richter
3	Siemens Industry Online Support	http://support.automation.siemens.com
\4\	Referenz auf den Beitrag	http://support.automation.siemens.com/WW/view/de/21043779
\5\	Visual Studio	http://www.microsoft.com
\6\	Plattformübergreif ende ILs	http://www.mono-project.com/
\7\	SIMATIC NET Industrielle Kommunikation Band 2	http://support.automation.siemens.com/WW/view/de/61630140

10 Historie

Tabelle 10-1

Version	Datum	Änderung
V1.0	05/2005	Erste Ausgabe
V2.0	12/2012	Komplette Überarbeitung auf STEP7 V11
V2.1	06/2014	Migration auf - STEP7 V13 - Visual Studio 2010
V2.2	07/2019	Aktualisierung STEP 7 V15.1