zenon
by COPA-DATA

zenon Treiber Handbuch

3S_V3

v.8.10

© 2019 Ing. Punzenberger COPA-DATA GmbH

Alle Rechte vorbehalten.

Die Weitergabe und Vervielfältigung dieses Dokuments ist - gleich in welcher Art und Weise - nur mit schriftlicher Genehmigung der Firma COPA-DATA gestattet. Technische Daten dienen nur der Produktbeschreibung und sind keine zugesicherten Eigenschaften im Rechtssinn. Änderungen - auch in technischer Hinsicht - vorbehalten.

Inhaltsverzeichnis

1 Willkommen bei der COPA-DATA Hilfe 5

2 3S_V3 5

3 3S_V3 - Datenblatt 7

4 Treiber-Historie 9

5 Voraussetzungen 10

5.1 PC 10

5.2 Steuerung 12

6 Konfiguration 12

6.1 Anlegen eines Treibers 13

6.2 Einstellungen im Treiberdialog 16

6.2.1 Allgemein 17

6.2.2 Einstellungen 21

6.2.3 Information 31

7 Variablen anlegen 32

7.1 Variablen im Editor anlegen 32

7.2 Adressierung 35

7.3 Treiberobjekte und Datentypen 37

7.3.1 Treiberobjekte 37

7.3.2 Zuordnung der Datentypen 39

7.4 Variablen anlegen durch Import 40

7.4.1 XML Import 40

7.4.2 DBF Import/Export 41

7.4.3 Online- und Offline-Import 47

7.5 Kommunikationsdetails (Treibervariablen) 53

8 Treiberspezifische Funktionen 59

8.1 Redundanz 59

8.2 Rückmeldung zur SPS 62

8.3 Byte-Reihenfolge 62

8.4 Exception Handling und Alarmierung 64

8.5 RDA 68

8.6 Echtzeitfähigkeit 70

9 Funktion Treiberkommandos 70

10 Fehleranalyse 75

10.1 Analysetool 75

10.2 Fehlernummern 76

10.3 Checkliste 78

1 Willkommen bei der COPA-DATA Hilfe

ZENON VIDEO-TUTORIALS

Praktische Beispiele für die Projektierung mit zenon finden Sie in unserem YouTube-Kanal (https://www.copadata.com/tutorial_menu). Die Tutorials sind nach Themen gruppiert und geben einen ersten Einblick in die Arbeit mit den unterschiedlichen zenon Modulen. Alle Tutorials stehen in englischer Sprache zur Verfügung.

ALLGEMEINE HILFE

Falls Sie in diesem Hilfekapitel Informationen vermissen oder Wünsche für Ergänzungen haben, wenden Sie sich per E-Mail an documentation@copadata.com.

PROJEKTUNTERSTÜTZUNG

Unterstützung bei Fragen zu konkreten eigenen Projekten erhalten Sie vom Support-Team, das Sie per E-Mail an support@copadata.com erreichen.

LIZENZEN UND MODULE

Sollten Sie feststellen, dass Sie weitere Module oder Lizenzen benötigen, sind unsere Mitarbeiter unter sales@copadata.com gerne für Sie da.

2 3S_V3

ALLGEMEIN

Der **3S_V3** Treiber verwendet für die Verbindung zur Steuerung den PLC-Handler von CoDeSys. Der PLC-Handler ist eine Schnittstelle zu Steuerungen mit Software-Versionen V2.3 und V3.

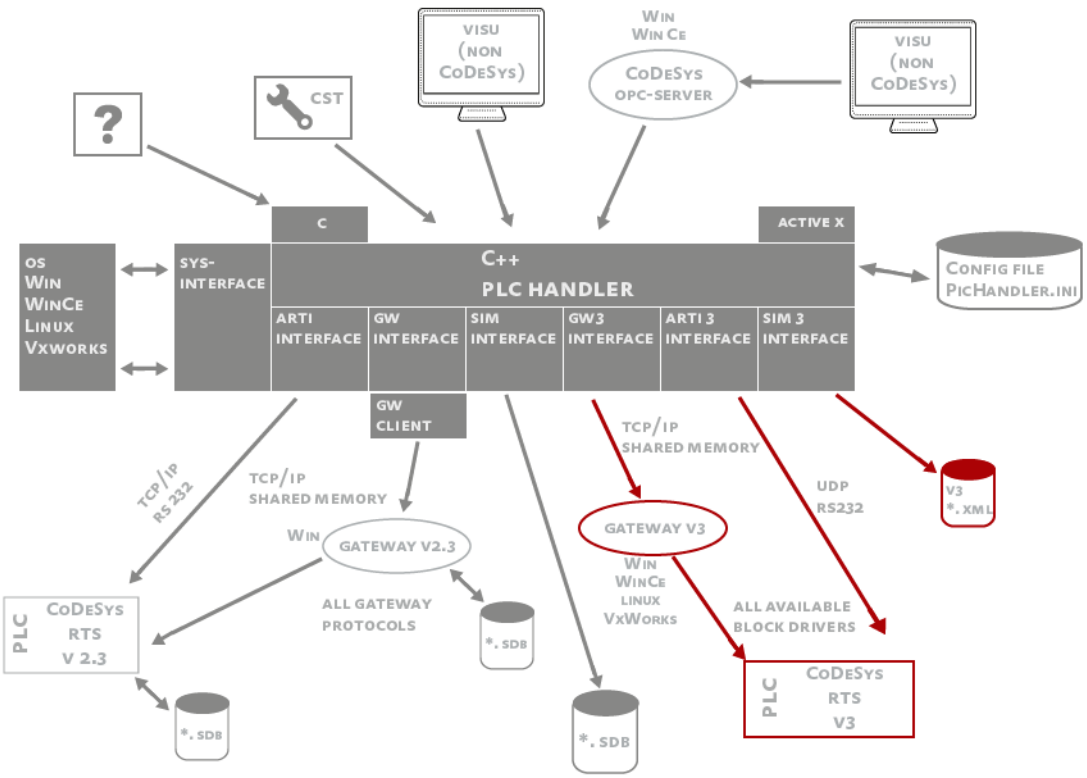
Info

Der Treiber unterstützt ab zenon 7.50 sowohl die Version 2.3 wie auch die Version 3 der CoDeSys-Software.

PLC-HANDLER

Unterschiedliche Versionen des PLC Handlers sind nicht kompatibel. Es muss als immer der zur jeweiligen Treiberversion passende PLC-Handler verwendet werden. PLC-Handler anderer Versionen müssen ersetzt werden. Details siehe Kapitel **PC** (auf Seite 10).

VERBINDUNG



Steuerungen der Version V3 und V2.3 werden über die sogenannte **Knotenadresse** angesprochen, die im erreichbaren Netzwerk eindeutig ist.

Die Verbindung zur Steuerung kann hergestellt werden:

- per ARTI (Asynchronous RunTime Interface) über die lokale Netzwerkkarte oder

- ein Gateway auf dem lokalen oder einem Remote Rechner

Der Treiber kann mehrere Steuerungen gleichzeitig ansprechen. Dabei wird für jede Verbindung eine eigene Instanz des PLC-Handlers angelegt.

SYMBOLISCHE ADRESSIERUNG

Der PLC-Handler spricht Variablen in der Steuerung nur über deren symbolischen Namen an. Die Adressierung über den Offset der Variablen ist nicht möglich. Allerdings kann bei komplexen Variablen (Strukturen, Arrays) der gesamte (binäre) Datenblock der Variable über den **Basisnamen** angesprochen werden. Das wird vom Treiber für RDA (auf Seite 59) benutzt.

In zenon Versionen bis 7.00 werden Variablen über die Variablenkennung adressiert. Ab Version 7.10 kann gewählt werden, ob die Kennung oder die Option **Symbolische Adresse** verwendet wird.

Damit der PLC-Handler die Variablen über den symbolischen Namen in der Steuerung findet, müssen Sie die im zenon Projekt verwendeten Variablen in der CoDeSys Projektiersoftware über ein Symbolkonfigurations-Objekt auswählen und in die Steuerung übertragen. Dabei wird gleichzeitig eine Symboldatei im XML-Format erzeugt. Diese kann für den Offline-Variablenimport verwendet werden.

Hinweis: Sind keine Symbole auf der Steuerung vorhanden, wird die Verbindung abgebrochen.

3 3S_V3 - Datenblatt

Allgemein:	
Treiberdateiname	3S_V3.exe
Treiberbezeichnung	3S v3 Treiber für PLC Handler
Steuerungs-Typen	3S CoDeSys v3 basierte Steuerungen; PAC Drive LMC von Schneider Electric. V2 Steuerungen ab Version V2.3.9.33
Steuerungs-Hersteller	3S; Festo; Elau; Schneider; Wago; Bosch Rexroth

Treiber unterstützt:	
Protokoll	3S-Arti; 3S-Gateway
Adressierung: Adress-basiert	Name based
Adressierung:	--

Treiber unterstützt:	
Namens-basiert	
Kommunikation spontan	--
Kommunikation pollend	X
Online Browsing	X
Offline Browsing	X
Echtzeitfähig	X
Blockwrite	--
Modemfähig	--
RDA numerisch	X
RDA String	--
Hysterese	--
erweiterte API	X
Unterstützung von Statusbit **WR-SUC**	X
alternative IP-Adresse	--

Voraussetzungen:	
Hardware PC	--
Software PC	PLCHandler von 3S; Die Datei 'PLCHandlerDll.dll' wird automatisch vom Setup mitinstalliert. Die Datei "GDrvABBTcpIpL2X.dll" wird benötigt wenn "TCP L2ABB" Gateway protocol für V2.3 verwendet wird. Diese Datei muss von ABB bezogen werden.
Hardware Steuerung	--
Software Steuerung	--
Benötigt v-dll	X

Plattformen:	
Betriebssysteme	Windows 10; Windows 7; Windows 8; Windows 8.1; Windows Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

4 Treiber-Historie

Datum	Build Nummer	Änderung
21.06.13	7.11.0.7498	Treiberdokumentation wurde neu erstellt
17.08.15	7.50.0.21651	Treiber unterstützt: ▸ Auswahl Herkunft Zeitstempel ▸ Block-Arrays ▸ einstellbares Präfix für importierte Variablennamen ▸ Exception-Handling
30.09.15	7.20.0.21830	Treiber unterstützt und benötigt PLC Handler 3.5.7.0.
10.1.2016	7.50.0.25589	Zusätzliche Unterstützung von CoDeSys Version 2.3.

TREIBERVERSIONIERUNG

Mit zenon 7.10 wurde die Versionierung der Treiber verändert. Ab dieser Version gibt es eine versionsübergreifende Build-Nummer. Das ist die Zahl an der 4. Stelle der Dateiversion.
Zum Beispiel: **7.10.0.4228** bedeutet: Der Treiber ist für Version **7.10**, Service Pack **0** und hat die Build-Nummer **4228**.

Erweiterungen oder Fehlerbehebungen werden zukünftig in einem Build eingebaut und sind dann ab der nächsthöheren Build-Nummer verfügbar.

Beispiel

Eine Treibererweiterung wurde in Build **4228** implementiert. Der Treiber, den Sie im Einsatz haben, verfügt über die Build-Nummer **8322**. Da die Build-Nummer Ihres Treibers höher ist als die Build-Nummer der Erweiterung, ist die Erweiterung enthalten. Die Versionsnummer des Treiber (die ersten drei Stellen der Dateiversion) spielen dabei keine Rolle. Die Treiber sind versionsunabsabhängig

5 Voraussetzungen

Dieses Kapitel enthält Informationen zu den Voraussetzungen, die für die Verwendung des Treibers erforderlich sind.

5.1 PC

ZUSÄTZLICHE SOFTWARE

PC

Der Treiber benötigt die **PLCHandlerDll.dll**, die von 3S hergestellt wird und das **PLCHandler-SDK** enthält. Die benötigte Version ist von der eingesetzten zenon Version abhängig:

ZENON 7.50

Ab Version 7.50 unterstützt zenon sowohl die Kommunikation von V3- und V2-Verbindungen.

Für eine Verbindung zur SPS mit Version 2 gelten folgende Regeln:

- Wird die Verbindung mit ARTI (Peer-to-Peer) hergestellt, ist keine zusätzliche Software erforderlich.
- Wird die Verbindung über ein Gateway hergestellt, müssen folgende Dateien zwingend in demselben Verzeichnis vorhanden sein, indem sich auch die Treiber-exe-Datei befindet:
  - GClient.dll
  - GDrvBase.dll
  - GHandle.dll
  - GHandleStdcall.dll
  - GSymbol.dll
  - GUtil.dll

  Hinweis: Das Verzeichnis kann auch mit der Umgebungsvariable *PATH* angegeben werden.
- Soll das Gateway auf dem lokalen Rechner automatisch (spätestens mit dem Treiber) gestartet werden, müssen für eine Verbindung zu V 2 folgende Dateien in demselben Verzeichnis vorhanden sein, indem sich auch die Client-DLLS (siehe oben) befinden:

- Gateway.exe
- GatewayDDE.exe
- CommSym.dll
- CommUsr.dll

- Die Version der Gateway Software sowie der DLLs ist 2.3.9.33 (oder höher).

KOMPATIBILITÄTSHINWEIS

Die Kommunikation zu Steuerungen mit Version 2-Software ist erst ab zenon 7.50 möglich. Für ältere Versionen beachten Sie folgende Voraussetzungen:

Ab zenon Version	Benötigte PLC-Handler Version	Anmerkungen
7.00	*3.5.3.60*	Die **PLCHandlerDll.dll** muss im Windows Ordner vorhanden sein.
7.11	*3.5.4.0*	Der PLC Handler wird im Windows Ordner installiert.
7.20 Build 21830	*3.5.7.0*	Ab zenon 7.20 wird die jeweils aktuelle Version der **PLCHandlerDll.dll** automatisch installiert. Der Installationsordner ist jetzt der selbe Ordner wie der des Treibers.

Info

Achten Sie darauf, dass SPS-Firmware und PLC-Handler dieselbe Versionsnummer haben. Nur so kann die korrekte Kommunikation gewährleistet werden.

Hinweis: Eine Übersicht über die am Rechner verwendeten Versionen finden Sie im Treiberdialog im Register Information.

CE

Für CE muss keine weitere Software installiert werden, das **PLCHandler-SDK** wird fest zum Treiber gelinkt.

Unter Windows CE ist es nicht möglich, mehrere Treiber vom gleichen Typ zu verwenden.

Achtung

CE wird nur bis Version 7.20 unterstützt.
Ab Version 7.50 müssen Runtime-Dateien für 7.20 erzeugt werden.

5.2 Steuerung

STEUERUNGEN

Der Treiber 3S_V3 dient zur Anbindung an eine SPS, auf der die 3S CoDeSys V3 oder V2.3 Software verwendet wird.

Beispiele: **3S CoDeSys**, **Control Win V3**, oder **Schneider LMC PacDrive**.

Die Verbindung zur Steuerung erfolgt wahlweise über

- die ARTI-Schnittstelle (*Asynchronous RunTime Interface*)
- ein lokales oder ein Remote Gateway.

Info

Ab zenon 7.50 werden vom Treiber sowohl Version 3 wie auch Version 2 der CoDeSys Software unterstützt.

6 Konfiguration

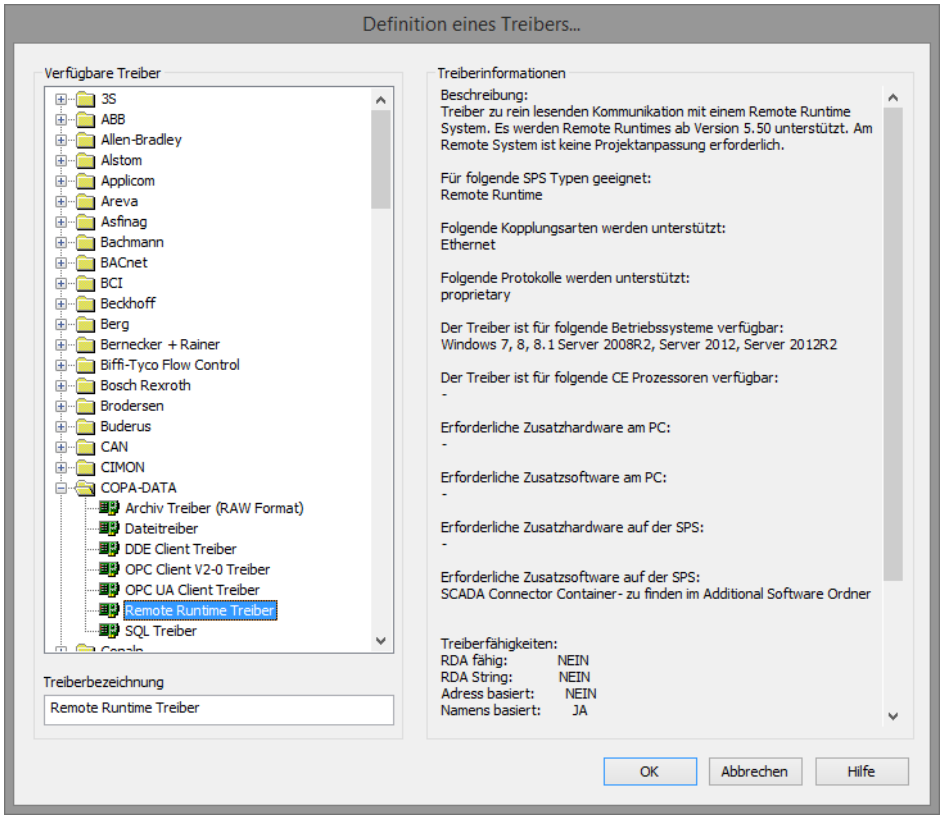
In diesem Kapitel lesen Sie, wie Sie den Treiber im Projekt anlegen und welche Einstellungen beim Treiber möglich sind.

Info

Weitere Einstellungen, die Sie für Variablen in zenon vornehmen können, finden Sie im Kapitel Variablen (main.chm::/15247.htm) der Online-Hilfe.

6.1 Anlegen eines Treibers

Im Dialog **Treiber erstellen** wählen Sie aus einer Liste jenen Treiber, den Sie neu anlegen wollen.



Parameter	Beschreibung
Verfügbare Treiber	Liste aller verfügbaren Treiber. Die Darstellung erfolgt in einer Baumstruktur: *[+]* erweitert die Ordnerstruktur und zeigt die darin enthaltenen Treiber. [-] reduziert die Ordnerstruktur Default: *keine Auswahl*
Treiberbezeichnung	Eindeutige **Bezeichnung** des Treibers. Default: *leer* Das Eingabefeld wird nach Auswahl eines Treibers aus der Liste der verfügbaren Treiber mit der vordefinierten **Bezeichnung** vorausgefüllt.
Treiberinformationen	Weiterführende Informationen über den gewählten Treiber. Default: *leer* Nach Auswahl eines Treibers werden in diesem

Parameter	Beschreibung
	Bereich die Informationen zum gewählten Treiber angezeigt.

DIALOG BEENDEN

Option	Beschreibung
OK	Übernimmt alle Einstellungen und öffnet den Treiberkonfigurationsdialog des ausgewählten Treibers.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

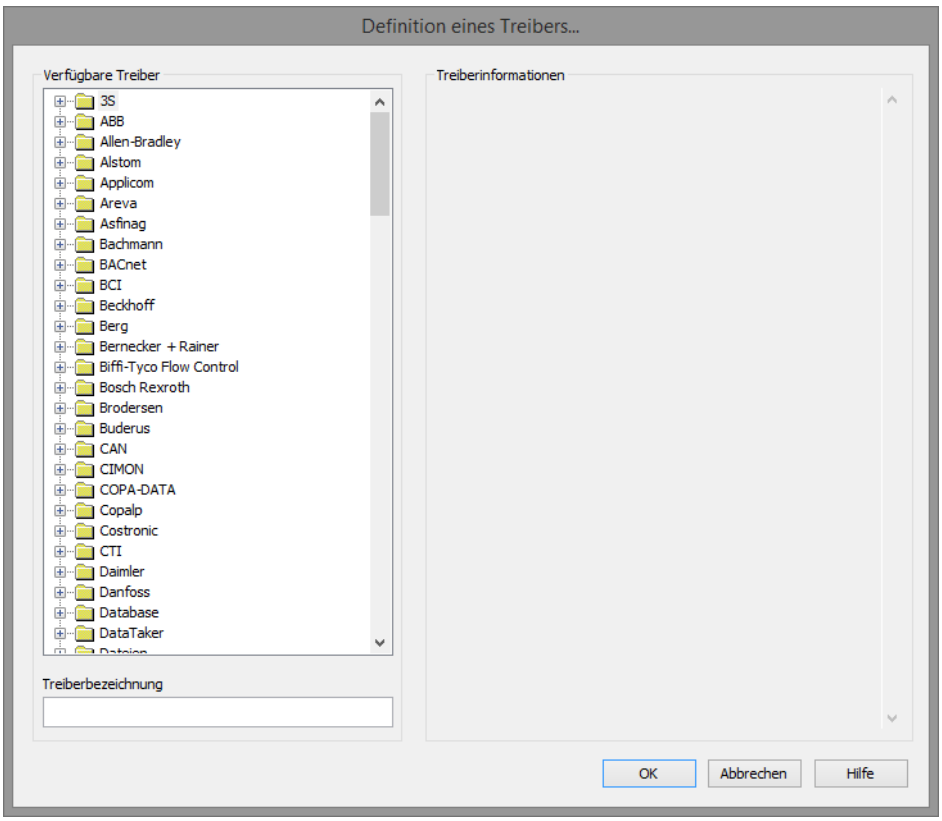
> **Info**
>
> Die Inhalte dieses Dialogs sind in der Datei Treiber_[Sprachkürzel].xml gespeichert. Sie finden diese Datei im Ordner *C:\ProgramData\COPA-DATA\zenon[Versionsnummer]*.

TREIBER NEU ANLEGEN

Um einen neuen Treiber anzulegen:

1. Klicken Sie mit der rechten Maustaste im Projektmanager auf **Treiber** und wählen Sie im Kontextmenü **Treiber neu** aus.
   Optional: Wählen Sie die Schaltfläche **Treiber neu** aus der Symbolleiste der Detailansicht der **Variablen**.
   Der Dialog **Treiber erstellen** wird geöffnet.

2. Der Dialog bietet eine Auflistung aller verfügbaren Treiber an.



3. Wählen Sie den gewünschten Treiber und benennen Sie diesen im Eingabefeld **Treiberbezeichnung**.
   Dieses Eingabefeld entspricht der Eigenschaft **Bezeichnung**. Per Default wird der Name des ausgewählten Treibers in diesem Eingabefeld automatisch eingefügt.
   Für die **Treiberbezeichnung** gilt:

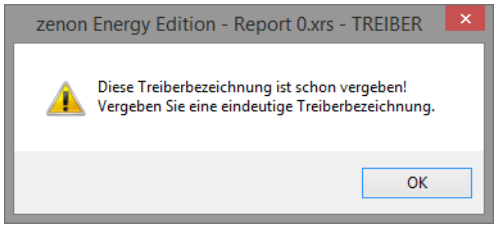
   - Die **Treiberbezeichnung** muss eindeutig sein.
     Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
     Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.
   - Die **Treiberbezeichnung** ist Bestandteil des Dateinamens.
     Daher darf Sie nur Zeichen enthalten, die vom Betriebssystem unterstützt werden. Nicht gültige Zeichen werden durch einen Unterstrich (_) ersetzt.
   - **Achtung:** Die Bezeichnung kann später nicht mehr geändert werden.

4. Bestätigen Sie den Dialog mit Klick auf die Schaltfläche **OK**.
   Der Konfigurationsdialog des ausgewählten Treibers wird geöffnet.

Hinweis: Treibernamen sind nicht sprachumschaltbar. Sie werden später immer in der Sprache angezeigt, in der sie angelegt wurden, unabhängig von der Sprache des Editors. Das gilt auch für Treiberobjekttypen.

DIALOG TREIBERBEZEICHNUNG BEREITS VORHANDEN

Ist ein Treiber bereits im Projekt vorhanden wird dies in einem Dialog angezeigt. Mit Klick auf die Schaltfläche **OK** wird der Warndialog geschlossen. Der Treiber kann korrekt benannt werden.



ZENON PROJEKT

Bei neu angelegten Projekten werden die folgenden Treiber automatisch angelegt:

- **Intern**
- **MathDr32**
- **SysDrv**

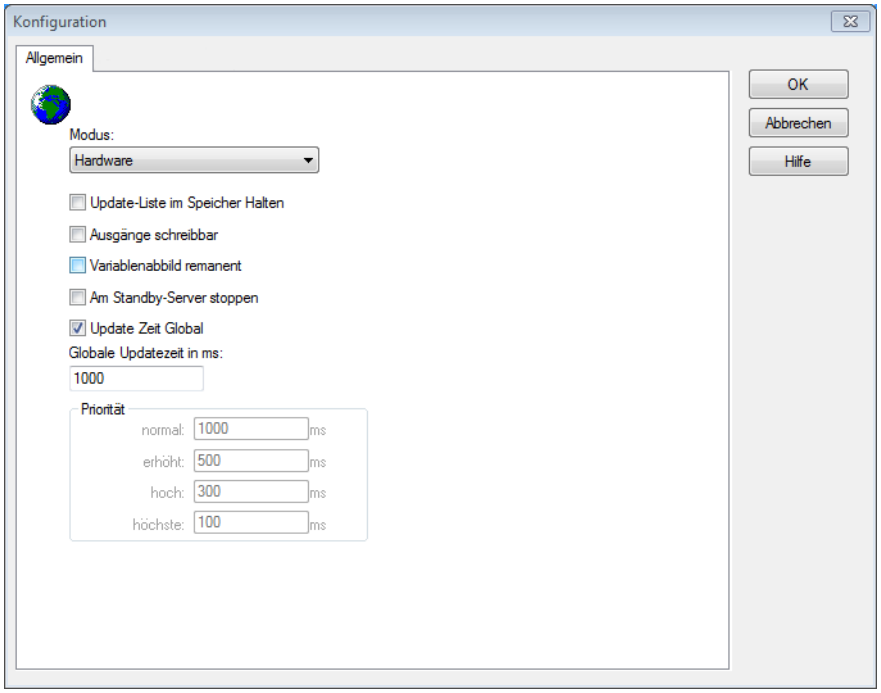
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein

Beim Anlegen eines Treibers wird der Konfigurationsdialog geöffnet. Um den Dialog später zum Bearbeiten zu öffnen, führen Sie einen Doppelklick auf den Treiber in der Liste aus oder klicken Sie auf die Eigenschaft **Konfiguration**.



Option	Beschreibung
Modus	Ermöglicht ein Umschalten zwischen Hardware und Simulationsmodus ▸ *Hardware*: Die Verbindung zur Steuerung wird hergestellt. ▸ Simulation - statisch: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus bleiben die Werte konstant oder die Variablen behalten die über zenon Logic gesetzen Werte. Jede Variable hat seinen eigenen Speicherbereich. Zum Beispiel zwei Variablen vom Typ Merker mit Offset 79, können zur Runtime unterschiedliche Werte haben und beeinflussen sich gegenseitig nicht. Ausnahme: Der Simulatortreiber. ▸ *Simulation - zählend*: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert.

Option	Beschreibung
	In diesem Modus zählt der Treiber die Werte innerhalb ihres Wertebereichs automatisch hoch. ▸ *Simulation - programmiert*: Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in einer in den Treiber integrierten zenon Logic Runtime ab. Details siehe Kapitel Treibersimulation. (main.chm::/25206.htm)
Update-Liste im Speicher Halten	Einmal angeforderte Variablen werden weiterhin von der Steuerung angefordert, auch wenn diese aktuell nicht mehr benötigt werden. Dies hat den Vorteil, dass z B. mehrmalige Bildumschaltungen nach dem erstmaligen Aufschalten beschleunigt werden, da die Variablen nicht neu angefordert werden müssen. Der Nachteil ist eine erhöhte Belastung der Kommunikation zur Steuerung.
Ausgänge schreibbar	▸ *Aktiv*: Ausgänge können beschrieben werden. ▸ *Inaktiv*: Das Beschreiben der Ausgänge wird unterbunden. **Hinweis**: Steht nicht für jeden Treiber zur Verfügungen.
Variablenabbild remanent	Diese Option speichert und restauriert den aktuellen Wert, den Zeitstempel und die Status eines Datenpunkts. Grundvoraussetzung: Die Variable muss einen gültigen Wert und Zeitstempel besitzen. Das Variablenabbild wird im Modus Hardware gespeichert, wenn einer dieser Status aktiv ist: ▸ Benutzerstatus *M1* (*0*) bis *M8* (*7*) ▸ *REVISION(9)* ▸ *AUS(20)* ▸ *ERSATZWERT(27)* Das Variablenabbild wird immer gespeichert wenn:

Option	Beschreibung
	▸ die Variable vom Objekttyp **Treibervariable** ist ▸ der Treiber im Simulationsmodus läuft. (nicht programmierte Simulation) Folgende Status werden beim Start der Runtime nicht restauriert: ▸ *SELECT(8)* ▸ *WR-ACK(40)* ▸ *WR-SUC(41)* Der Modus **Simulation - programmiert** beim Treiberstart ist kein Kriterium, um das remanente Variablenabbild zu restaurieren.
Am Standby Server stoppen	Einstellung für Redundanz bei Treibern, die nur eine Kommunikationsverbindung erlauben. Dazu wird der Treiber am Standby Server gestoppt und erst beim Hochstufen wieder gestartet. **Achtung:** Ist diese Option aktiv, ist die lückenlose Archivierung nicht mehr gewährleistet. ▸ *Aktiv*: Versetzt den Treiber am nicht-prozessführenden Server automatisch in einen Stopp-ähnlichen Zustand. Im Unterschied zum Stoppen über Treiberkommando erhält die Variable nicht den Status **abgeschaltet** (**statusverarbeitung.chm::/24150.htm**), sondern einen leeren Wert. Damit wird verhindert, dass beim Hochstufen zum Server nicht relevante Werte in AML , CEL und Archiv erzeugt werden. Default: *inaktiv* **Hinweis:** Nicht verfügbar, wenn CE Terminal als Datenserver dient. Weitere Informationen dazu erhalten Sie im Handbuch zenon Operator im Kapitel CE Terminal als Datenserver.
Update Zeit Global	Einstellung für globale Update-Zeiten in Millisekunden: ▸ *Aktiv*: Die eingestellte **Globale Update Zeit** wird für alle Variablen im Projekt verwendet. Die bei den

Option	Beschreibung
	Variablen eingestellte Priorität wird nicht verwendet. ▸ *Inaktiv*: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet. **Ausnahmen:** Spontane Treiber ignorieren diese Option. Sie nutzen in der Regel die kürzest mögliche Update Zeit. Details siehe Abschnitt **Update Zeit spontane Treiber**.
Priorität	Hier werden die Pollingzeiten der einzelnen Prioritätsklassen eingestellt. Alle Variablen mit der entsprechenden Priorität werden in der eingestellten Zeit gepollt. Die Zuordnung der Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften. Mit den Prioritätsklassen kann die Kommunikation der einzelnen Variablen auf die Wichtigkeit oder benötigte Aktualität abgestuft werden. Daraus ergibt sich eine verbesserte Verteilung der Kommunikationslast. **Achtung:** Prioritätsklassen werden nicht von jedem Treiber unterstützt, z.B. von spontan kommunizierenden zenon Treibern.

DIALOG BEENDEN

Option	Beschreibung
OK	**Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.**
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

UPDATE ZEIT SPONTANE TREIBER

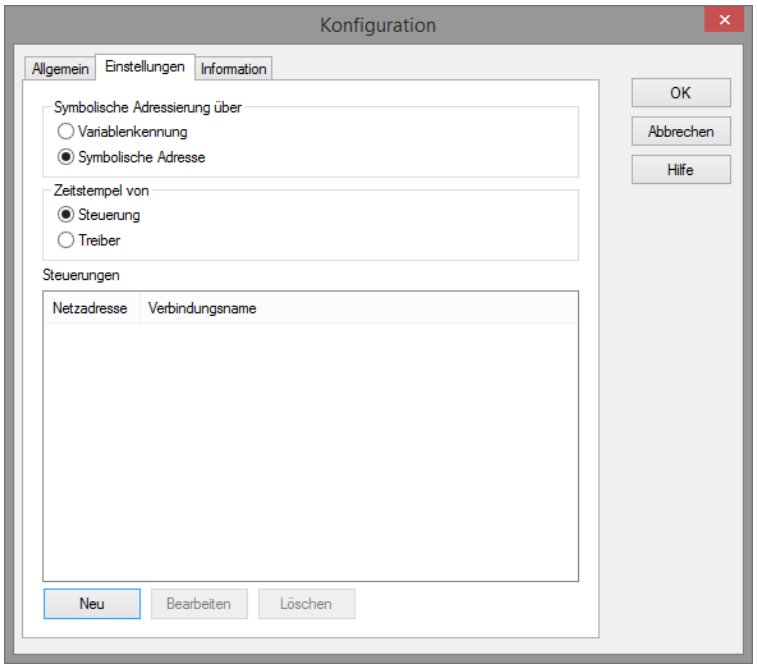
Bei spontanen Treibern wird beim **Sollwert Setzen**, **Advisen** von Variablen und bei **Requests** sofort ein Lesezyklus ausgelöst - unabhängig von der eingestellten Update Zeit. Damit wird sicher gestellt, dass der Wert nach dem Schreiben in der Visualisierung sofort zur Verfügung steht. In der Regel beträgt die Updatezeit 100 ms.

Spontane Treiber sind **ArchDrv**, **BiffiDCM**, **BrTcp32**, **DNP3**, **Esser32**, **FipDrv32**, **FpcDrv32**, **IEC850**, **IEC870**, **IEC870_103**, **Otis**, **RTK9000**, **S7DCOS**, **SAIA_Slave**, **STRATON32** und **Trend32**.

6.2.2 Einstellungen

In der Registerkarte Einstellungen konfigurieren Sie:

- die Art der symbolischen Adressierung der Variablen
- Herkunft des Zeitstempels
- Verbindungen zur SPS



Parameter	Beschreibung
Symbolische Adressierung über	Optionsfeld zur Auswahl, welche Quelle für die symbolische Adressierung der Variablen verwendet wird: ▸ **Variablenkennung** Symbolische Adressierung der Variablen erfolgt über die Variablenkennung. ▸ **Symbolische Adresse** Symbolische Adressierung der Variablen erfolgt über die Symbolische Adresse. **Hinweis:** Diese Auswahl ist ab der zenon Version 7.10 möglich. Bis einschließlich Version 7.00 wird immer die Variablenkennung verwendet. In diesem Fall werden die Optionen

Parameter	Beschreibung
	Variablenkennung und **Symbolische Adresse** nicht angezeigt.
Zeitstempel von	Optionsfeld zur Auswahl, wie der Zeitstempel neuer Variablenwerte ermittelt wird: ▸ **Steuerung**: Zeitstempel der Steuerung wird verwendet ▸ **Treiber**: Zeitstempel des Treibers wird verwendet

STEUERUNGEN

Die Verbindungen werden durch Klick auf die Schaltfläche **Neu** im Dialog zur Konfiguration einer Verbindung (auf Seite 24) projektiert. Bestehende Verbindungskonfigurationen können mit Auswahl und Klick auf die Schaltfläche **Bearbeiten** geändert werden.

Parameter	Beschreibung
Steuerungen	Liste der konfigurierten Verbindungen. ▸ *Netzadresse:* Netzadresse der Verbindung, wie für die Verbindung konfiguriert. Die Netzadresse muss eindeutig sein. ▸ *Verbindungsname:* Name der Verbindung, wie für die Verbindung konfiguriert. Dieser Verbindungsname entspricht der konfigurierten Knotenadresse (auf Seite 24) im Dialog (auf Seite 24) zur Konfiguration einer Verbindung. Die Spalten der Liste können bei gedrückter Maustaste durch Klick auf den Spaltenrand in der Breite angepasst werden.
Neu	Öffnet den Dialog (auf Seite 24) zur Konfiguration einer Verbindung.
Bearbeiten	Öffnet für die ausgewählte Verbindung den Dialog (auf Seite 24) zur Konfiguration einer Verbindung.
Löschen	Löscht die ausgewählte Verbindung nach einer Sicherheitsabfrage.

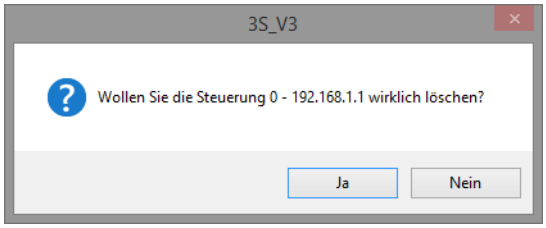
DIALOG BEENDEN

Parameter	Beschreibung
OK	Übernimmt alle Änderungen in allen Registerkarten und schließt

Parameter	Beschreibung
	den Dialog.
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

SICHERHEITSABFRAGE BEI LÖSCHEN EINER VERBINDUNG

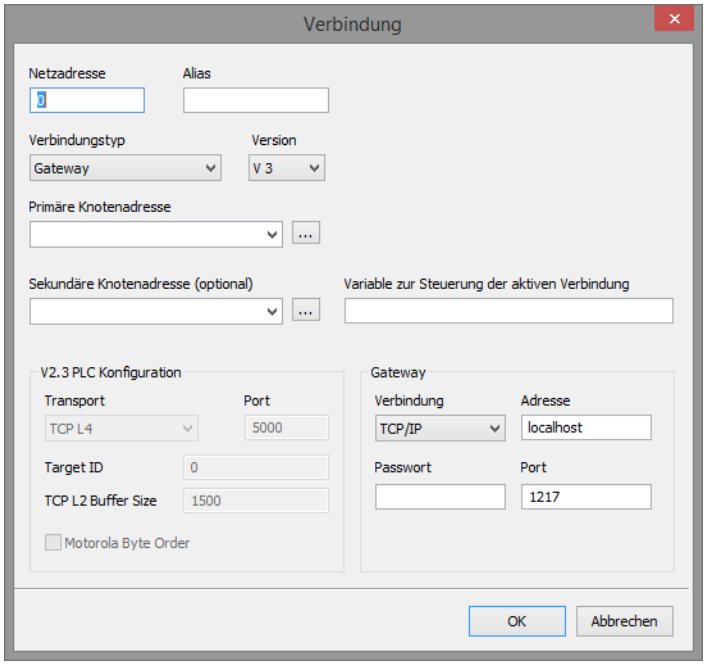
Wird eine Steuerung zum Löschen ausgewählt, wird vor dem Löschen eine Sicherheitsabfrage angezeigt.



Parameter	Beschreibung
Ja	Die Verbindung wird aus der Liste der konfigurierten Verbindungen (**Steuerungen**) entfernt.
Nein	Die Verbindung wird nicht gelöscht und verbleibt in der Liste der konfigurierten Verbindungen. Der Dialog wird geschlossen.

6.2.2.1 Verbindung

Konfiguration der Verbindung zu einer Steuerung.



Parameter	Beschreibung
Netzadresse	Eingabe der Netzadresse. Die Netzadresse muss eindeutig sein und darf nicht doppelt verwendet werden. Ist eine Netzadresse bereits vergeben, wird dies durch einen Hinweisdialog angezeigt. Die Eingabe wird bei Klick auf die Schaltfläche **OK** überprüft. Im Fehlerfall wird ein Warndialog angezeigt. **Achtung:** Wird eine numerische Zahl ausserhalb des Wertebereichs eingegeben, wird die Netzadresse mit dem Wert *-1* projektiert. Default: Als Defaultwert wird die niedrigste, noch nicht vergebene Netzadresse vorgegeben. Diese Netzadresse entspricht der zenon Eigenschaft **Netzadresse**.
Alias	Für jede Verbindung kann ein Alias angegeben werden. Dieser dient als Präfix für die Namen der Variablen, die aus dem Treiber importiert werden. Ist der Alias leer, wird die Knotenadresse als Präfix verwendet. Maximale Länge: *8 Zeichen*

Parameter	Beschreibung
	Hinweis: Die Eingabe von mehr als 8 Zeichen ist nicht möglich.
Verbindungstyp	Auswahl des Verbindungstyps aus Dropdownliste: ▸ *Gateway:* Die Kommunikation zur SPS erfolgt über ein Gateway auf dem lokalen oder einem Remote Rechner Weitere Konfigurationsmöglichkeiten für den Verbindungstyp Gateway finden Sie in der Konfigurationsgruppe **Gateway**. ▸ *ARTI*: direkte Kommunikation zur SPS über die lokale Netzwerkkarte (Peer-to-Peer Verbindung) Default: *Gateway*
Version	Auswahl der zu verwendenden Version aus Dropdownliste: ▸ V3 ▸ V2.3 Weitere Konfigurationsmöglichkeiten bei der Verwendung von V2.3 finden Sie in der Konfigurationsgruppe **V2.3 PLC Konfiguration**. Default: *V3*
Primäre Knotenadresse	Knotenadresse oder Knotenname der Steuerung. Für redundante Kommunikation: Knotenadresse oder Knotenname der primären Steuerung. Eingabe von Knotenadresse oder Knotenname in das Feld oder Auswahl aus Dropdownliste. Klick auf die Schaltfläche **...** öffnet den Dialog (auf Seite 30) zur Auswahl einer Steuerung. **Achtung:** Diese Suche nach einer Steuerung steht für V2.3-Kommunikation nicht zur Verfügung! Das Durchsuchen des Netzwerks ist mit V2.3 nicht möglich. Beim Öffnen wird im Netzwerk nach Steuerungen gesucht, die über die gewählte Verbindungsart erreichbar sind. Ist die Suche nach einer Steuerung nicht möglich, da sich Projektierungsrechner und Steuerung in unterschiedlichen Netzwerken befinden, muss die Adresse manuell eingegeben werden. **Redundanz:** Der Treiber verbindet sich per default mit der primären Verbindung. Kann

Parameter	Beschreibung
	diese nicht hergestellt werden oder fällt aus, wird versucht auf die sekundäre Verbindung umzuschalten. Die primäre Verbindung hat dabei immer höhere Priorität. Sind beide verfügbar, erfolgt die Kommunikation immer über die primäre Adresse. Beachten Sie: Dabei werden alle Variablen immer nur bei der aktiven Verbindung angemeldet. Werte werden nur von der aktiven Verbindung gelesen. Werte werden aber immer auf beide Steuerungen geschrieben, wenn die Verbindungen verfügbar sind. **Knotenadresse und Knotenname** ▸ Die Knotenadresse kann für unterschiedliche Verbindungsarten unterschiedlich lauten. Sie ist aber für die gewählte Verbindungsart immer eindeutig. **Hinweis:** Eventuell erhält eine Steuerung bei einem Neustart eine neue Knotenadresse, da die ursprüngliche als vergeben geführt wird. ▸ Der Knotenname ist nicht eindeutig. Zum Beispiel können auf einem Gerät mehrere Steuerungen laufen. Diese können dann nur über die Knotenadresse identifiziert werden.
Sekundäre Knotenadresse	Knotenadresse oder Knotenname der redundanten Steuerung. Eingabe einer Knotenadresse oder eines Knotennamens in das Feld oder Auswahl aus Dropdownliste. Klick auf die Schaltfläche **...** öffnet den Dialog (auf Seite 30) zur Auswahl einer Steuerung. **Hinweis:** Die sekundäre Knotenandresse kann auch leer bleiben. In diesem Fall wird bei einem Verbindungsausfall der primären Verbindung keine Redundanzumschaltung durchgeführt.
Variable zur Steuerung der aktiven Verbindung	Name der Variable mit der die Verbindung der Kommunikation gewählt wird. Diese Variable muss vom Treiberobjekttyp **Verbindungsstatus** im zenon Editor projektiert sein. Beachten Sie: ▸ Geben Sie hier die symbolische Adresse der Variable an. Dies entspricht der Variableneigenschaft **Symbolische Adresse** im <CD_PRODUCDTNAME> Editor. Prinzipiell genügt es, wenn diese Variable auf beiden Steuerungen konfiguriert ist. ▸ Die Variable muss vom Datentyp *BOOL* sein.

Parameter	Beschreibung
	Ist die sekundäre Knotenadresse leer, hat diese Option keine Bedeutung.

V2.3 PLC KONFIGURATION

Konfigurationen für die Kommunikation mit dem 3S V3-Treiber zu einer SPS mit V 2.3-Software.

Ist in der Eigenschaft **Version** V3 ausgewählt, sind die Eigenschaften der V2.3 PLC Konfiguration ausgegraut.

Parameter	Beschreibung
Transport	Dropdownliste zur Auswahl des Kommunikationsprotokolls. ▸ *TCP L4* ▸ *TCP L2 Route* ▸ *TCP L2* ▸ *TCP L2ABB* **Hinweis:** Unterstützt nur Steuerungen des Herstellers ABB. Für die Konfiguration muss die **GDrvABBTcpIpL2X.dll** am System vorhanden sein und sich im selben Ordner wie die **Gateway.exe** befinden. Die DLL muss direkt beim Hersteller ABB bezogen werden. Weitere Hinweise siehe auch **Checkliste** (auf Seite 78). Default: *TCP L4*
Port	Portnummer der Kommunikation. Default: *5000* Die Eingabe wird bei Klick auf die Schaltfläche **OK** überprüft. Im Fehlerfall wird ein Warndialog angezeigt.
Target ID	Eindeutige ID des Zielgeräts. Default: *0* Die Eingabe wird bei Klick auf die Schaltfläche **OK** überprüft. Im Fehlerfall wird ein Warndialog angezeigt.
TCP L2 Buffer Size	Puffergröße des TCP L2-Protokolls.

Parameter	Beschreibung
	Wird nur für die TCP L2-Kommunikationsprotokolle verwendet. Für TCP L4 hat diese Eigenschaft keine Auswirkung. Default: *1500* Die Eingabe wird bei Klick auf die Schaltfläche **OK** überprüft. Im Fehlerfall wird ein Warndialog angezeigt.
Motorola Byte Order	Byte-Reihenfolge für den Empfang oder das Senden von Daten ▸ *aktiviert*: Daten werden mit dem Big-Endian (Motorola) Format verarbeitet. ▸ *nicht aktiviert*: Daten werden im Little-Endian (Intel) Format verarbeitet. **Hinweis:** Die Byte-Reihenfolge wird auch auf Basis des RDA Typs automatisch erkannt. Weitere Informationen dazu finden Sie im Kapitel Byte-Reihenfolge (auf Seite 62).

GATEWAY

Konfiguration der Verbindung zum Gateway.

Ist in der Eigenschaft **Verbindungstyp ARTI** ausgewählt, sind die Eigenschaften der Gateway-Konfiguration ausgegraut.

Parameter	Beschreibung
Verbindung	Auswahl der Verbindungsart vom PLC-Handler zum Gateway aus Dropdownliste: ▸ *TCP/IP*: TCP/IP-Verbindung ▸ *Shared Memory*: Nutzung des Shared Memory am lokalen Rechner Default: *TCP/IP*
Adresse	Eingabefeld für die Adresse des Gateways.

Parameter	Beschreibung
	Die Adresse kann sowohl numerisch (192.168.1.1) wie auch alphanumerisch (z.B.: *localhost*) konfiguriert werden. Eine Überprüfung der Gültigkeit einer Eingabe wird nicht durchgeführt. **Hinweis:** nur verfügbar, wenn als **Verbindung** *TCP/IP* konfiguriert ist.
Passwort	Passwort des Gateways (optional). **Hinweis:** nur verfügbar, wenn als **Verbindung** *TCP/IP* konfiguriert ist.
Port	Port des Gateways. **Hinweis:** nur verfügbar, wenn als **Verbindung** *TCP/IP* konfiguriert ist. Default: *1217*

KOMMUNIKATIONSEINSTELLUNGEN

Erlaubt die Konfiguration des Kommunikationsverhaltens einer Verbindung.

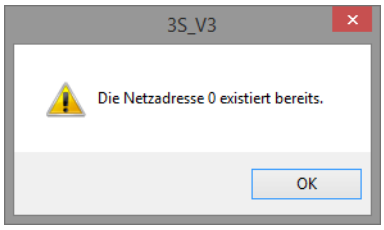
Parameter	Beschreibung
Timeout	Definiert das Timeout der Kommunikation in ms. Erlaubte Einstellungen für dieses Timeout liegen im Bereich von 1ms bis 1000000ms. Default: 5000ms
Wiederholversuche	Definiert die Anzahl der Wiederholversuche beim Empfang eingehender Daten bevor ein COMM_FATAL Fehler signalisiert wird. Erlaubte Einstellungen für diesen Wert liegen im Bereich von 1 bis 10. Default: 3

DIALOG BEENDEN

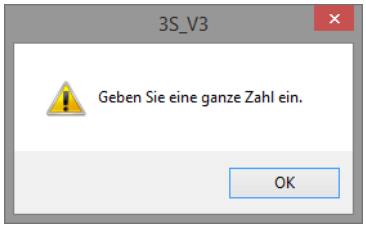
Parameter	Beschreibung
OK	Übernimmt Einstellungen und schließt den Dialog.
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.

WARNDIALOG BEI FEHLEINGABE

Netzadresse bereits vorhanden:

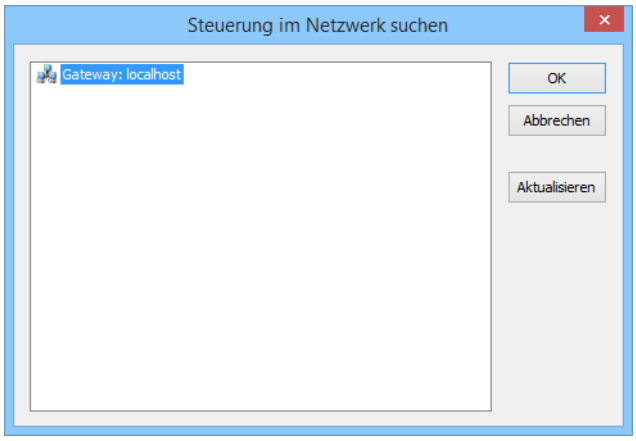


Wird eine ungültige Eingabe projektiert (z.B. Buchstaben) wird dies ebenfalls in einem Warndialog angezeigt.



6.2.2.1.1 Steuerung im Netzwerk suchen

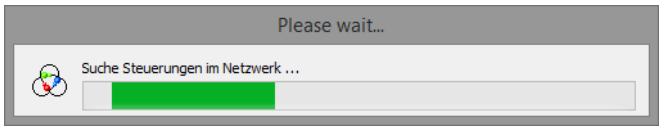
Mit Klick auf die Schaltfläche **Knotenadresse** wird der Dialog für die Auswahl einer Steuerung geöffnet. Beim Öffnen wird nach Steuerungen gesucht. Gefundene Steuerungen werden zur Auswahl angeboten.



Parameter	Beschreibung
Liste der Steuerungen	Listet alle gefundenen Steuerungen auf. Bei einer Verbindung via Gateway wird die im Dialog **Verbindung** konfigurierte Adresse als Hauptknoten angezeigt.
OK	Übernimmt ausgewählte Steuerung und schließt den Dialog.

Parameter	Beschreibung
Abbrechen	Verwirft Auswahl und schließt den Dialog.
Aktualisieren	Sucht erneut nach Steuerungen.

DURCHSUCHEN DES NETZWERK

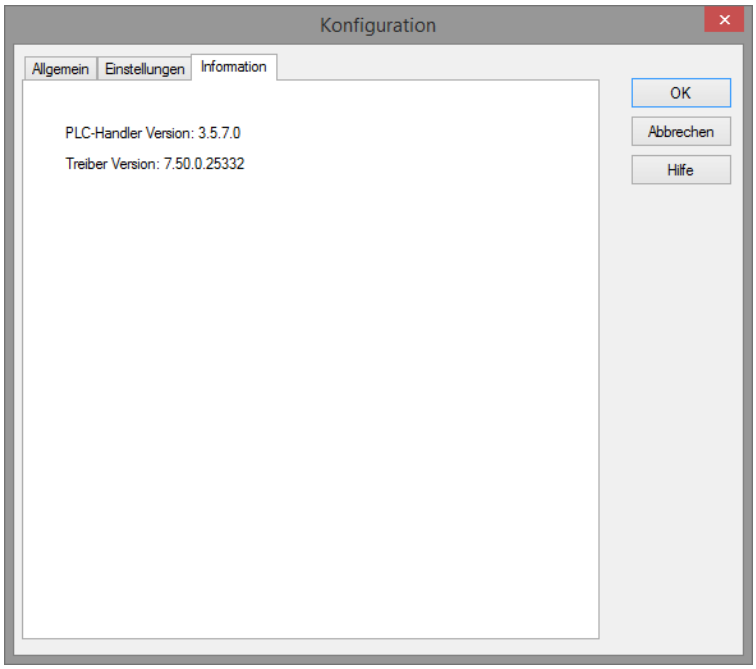


Das Durchsuchen des Netzwerks kann - je nach Ausprägung und Größe des Netzes - einige Zeit in Anspruch nehmen. Während dieser Zeit wird ein Dialog mit einer Fortschrittsanzeige visualisiert.

6.2.3 Information

Informationen zu den Versionen von:

- PLC-Handler
- Treiber



7 Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

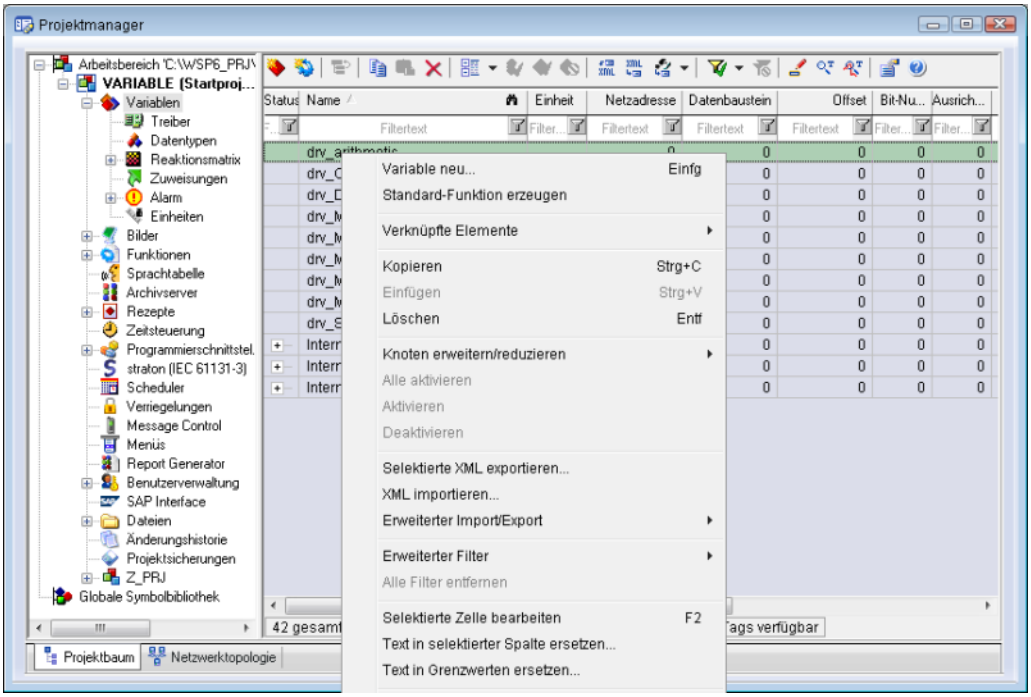
Variablen können angelegt werden:

- als einfache Variable
- in Arrays (main.chm::/15262.htm)
- als Struktur-Variablen (main.chm::/15278.htm)

DIALOG VARIABLE

Um eine neue Variable zu erstellen, gleich welchen Typs:

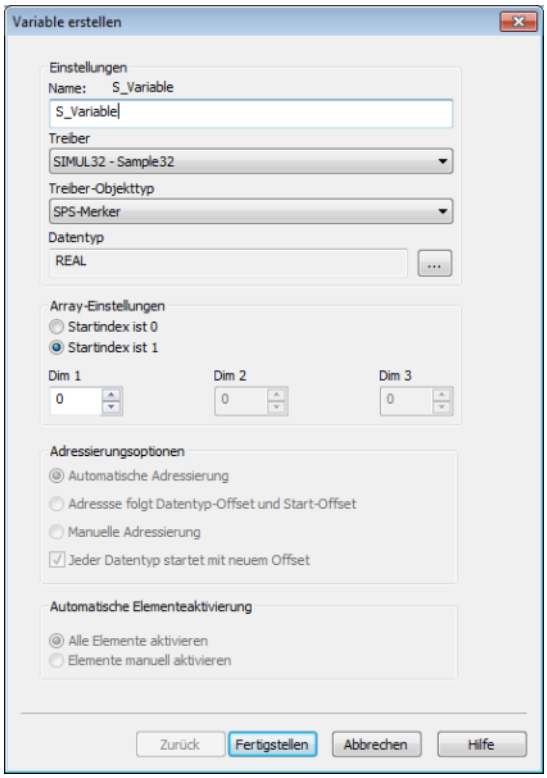
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



   Der Dialog zur Konfiguration der Variable wird geöffnet.

2. Konfigurieren Sie die Variable.
3. Welche Einstellungen möglich sind, hängt ab vom Typ der Variablen.

DIALOG VARIABLE ERSTELLEN



Eigenschaft	Beschreibung
Name	Eindeutiger Name der Variablen. Ist eine Variable mit gleichem Namen im Projekt bereits vorhanden, kann keine weitere Variable mit diesem Namen angelegt werden. Maximale Länge: 128 Zeichen **Achtung:** Die Zeichen **#** und **@** sind für Variablennamen nicht erlaubt. Bei Verwendung nicht zugelassener Zeichen kann die Variablenerstellung nicht abgeschlossen werden, die Schaltfläche **Fertigstellen** bleibt inaktiv. **Hinweis:** Manche Treiber erlauben die Adressierung auch über die Eigenschaft **Symbolische Adresse**.
Treiber	Wählen Sie aus der Dropdownliste den gewünschten Treiber. **Hinweis:** Sollte im Projekt noch kein Treiber angelegt sein, wird automatisch der Treiber für interne Variable (**Intern.exe** (**Main.chm::/Intern.chm::/Intern.htm**)) geladen.
Treiberobjekttyp (**cti.chm::/28685.htm**)	Wählen Sie aus der Dropdownliste den passenden Treiberobjekttyp aus.

Eigenschaft	Beschreibung
Datentyp	Wählen Sie den gewünschten Datentyp. Klick auf die Schaltfläche ... öffnet den Auswahl-Dialog.
Array-Einstellungen	Erweiterte Einstellungen für Array-Variablen. Details dazu lesen Sie im Abschnitt Arrays.
Adressierungsoptionen	Erweiterte Einstellungen für Arrays und Struktur-Variablen. Details dazu lesen Sie im jeweiligen Abschnitt.
Automatische Elementeaktivierung	Erweiterte Einstellungen für Arrays und Struktur-Variablen. Details dazu lesen Sie im jeweiligen Abschnitt.

SYMBOLISCHE ADRESSE

Die Eigenschaft **Symbolische Adresse** kann für die Adressierung alternativ zu **Name** oder **Kennung** der Variablen verwendet werden. Die Auswahl erfolgt im Treiberdialog, die Konfiguration in der Variableneigenschaft. Beim Import von Variablen unterstützter Treiber wird die Eigenschaft automatisch eingetragen.

Maximale Länge: *1024* Zeichen.

Folgende Treiber unterstützen die **Symbolische Adresse**:

- **3S_V3**
- **AzureDrv**
- **BACnetNG**
- **IEC850**
- **KabaDPServer**
- **OPCUA32**
- **Phoenix32**
- **POZYTON**
- **RemoteRT**
- **S7TIA**
- **SEL**
- **SnmpNg32**
- **PA_Drv**

ABLEITUNG VOM DATENTYP

Messbereich, **Signalbereich** und **Sollwert Setzen** werden immer:

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

Hinweis Signalbereich: Bei einem Wechsel auf einen Datentyp, der den eingestellten **Signalbereich** nicht unterstützt, wird der **Signalbereich** automatisch angepasst. Zum Beispiel wird bei einem Wechsel von **INT** auf **SINT** der **Signalbereich** auf *127* geändert. Die Anpassung erfolgt auch dann, wenn der **Signalbereich** nicht vom Datentyp abgeleitet wurde. In diesem Fall muss der **Messbereich** manuell angepasst werden.

7.2 Adressierung

Gruppe/Eigenschaft	Beschreibung
Allgemein	Gruppe mit allgemeinen Eigenschaften.
Name	Frei vergebbarer Name. **Achtung:** Je zenon Projekt muss der Name eindeutig sein.
Kennung	▸ Bis zenon Version 7.00: **Symbolische Adresse** ▸ Ab Version 7.10: Wahlweise **Symbolische Adresse** oder frei vergebbare Kennung, z. B. für Betriebsmittelkennung, Kommentar usw ...
Adressierung	Gruppe mit Eigenschaften für die Adressierung.
Netzadresse	Netzadresse der Variablen. Diese Adresse bezieht sich auf die Netzadresse der Verbindungsprojektierung im Treiber. Damit wird ausgewählt auf welcher Steuerung sich die Variable befindet.
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Wird für diesen Treiber nicht verwendet.
Ausrichtung	Wird für diesen Treiber nicht verwendet.
Bitnummer	Wird für diesen Treiber nicht verwendet.
Stringlänge	Nur verfügbar bei String-Variablen. Maximale Anzahl von Zeichen, die die Variable aufnehmen kann.
Symbolische Adresse	Die Eigenschaft **Symbolische Adresse** kann für die Adressierung alternativ zu **Name** oder **Kennung** der Variablen verwendet werden. Die Auswahl erfolgt im Treiberdialog, die Konfiguration in der Variableneigenschaft.

Gruppe/Eigenschaft	Beschreibung
	Beim Import von Variablen unterstützter Treiber wird die Eigenschaft automatisch eingetragen. Maximale Länge: *1024* Zeichen.
Treiber Anbindung/Treiberobjekttyp	Objekttyp der Variablen. Wird abhängig vom verwendeten Treiber beim Erstellen der Variablen ausgewählt und kann hier geändert werden.
Treiber Anbindung/Datentyp	Datentyp der Variablen. Wird beim Erstellen der Variablen ausgewählt und kann hier geändert werden. **ACHTUNG:** Wenn der Datentyp nachträglich geändert wird, müssen alle anderen Eigenschaften der Variablen überprüft bzw. angepasst werden.
Treiber Anbindung/Priorität	Einstellen der Prioritätsklasse. Damit wird die Variable der Prioritätsklasse zugeordnet, so wie sie im Treiberdialog in der Registerkarte **Allgemein** konfiguriert wurde. Die Prioritätsklassen werden nur dann verwendet, wenn die **Update Zeit Global** deaktiviert ist. Bei aktivierter Option **Update Zeit Global** und Verwendung der Prioritätsklassen erfolgt ein Fehlereintrag in die Log-Datei des Systems. Der Treiber verwendet die höchstmögliche Priorität.

BLOCK-ARRAYS

Block-Arrays können gelesen, geschrieben und importiert werden. Es wird der Symbolname ohne Indizies benutzt. Das Zeichen **#** ist nicht zugelassen.
Beispiel: **MAIN.MyArray**.

Die Anzahl des Indizies muss mit der in der SPS übereinstimmen.
Beispiel: **For MyArray: ARRAY [0..99] OF INT;**

Hinweis: In den Variableneigenschaften von zenon muss in der Gruppe **Zusätzliche Einstellungen** die Eigenschaft **Größe Blockarray** auf den Wert *100* gesetzt werden.

Achtung

Beachten Sie: In zenon wird mit *0* zu zählen begonnen, in der SPS aber mit *1*.

7.3 Treiberobjekte und Datentypen

Treiberobjekte sind in der Steuerung verfügbare Bereiche wie z. B. Merker, Datenbausteine usw. Hier lesen Sie, welche Treiberobjekte vom Treiber zur Verfügung gestellt werden und welche IEC-Datentypen dem jeweiligen Treiberobjekt zugeordnet werden können.

7.3.1 Treiberobjekte

Folgende Objekttypen stehen in diesem Treiber zur Verfügung:

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Beschreibung
Alarm Event	64	X	X	*INT, USINT*	Wird für Exception Handling (auf Seite 59) verwendet.
Alarm Variable	65	X	--	*BOOL, DINT, STRING, UDINT*	Wird für Exception Handling (auf Seite 59) verwendet.
Verbindungsstatus	43	X	--	*BOOL, USINT*	Projektierte Variable (Verbindung) gibt an, welche Verbindung aktiv ist, wenn beide Verbindungen verfügbar sind. ▸ *0: Primäre Verbindung* ▸ *1: Sekundäre Verbindung* ▸ *3: Bitwert 0* ▸ *4: Bitwert 1* Beachten Sie die dazu die weiteren Informationen im Kapitel Redundanz (auf Seite 59).
SPS-Merker	8	X	X	*BOOL, USINT, SINT, BYTE, UINT, WORD, INT, UDINT,*	

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Beschreibung
				DWORD, DINT, ULINT, LINT, LWORD, REAL, LREAL, STRING, WSTRING, DT, TOD, DATE, TIME	
Status Feedback	66	X	--	*DINT*	Wird für Exception Handling (auf Seite 59) verwendet.
Kommunikationsdetails	35	X	X	*BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING*	Variablen für die statische Analyse der Kommunikation; wird zwischen Treiber und Runtime übertragen (nicht zur SPS). **Hinweis**: Die Adressierung und das Verhalten ist bei den meisten zenon Treibern gleich. Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 53).

Legende:

X: wird unterstützt

--: wird nicht unterstützt

7.3.2 Zuordnung der Datentypen

Alle Variablen in zenon werden von IEC-Datentypen abgeleitet. In folgender Tabelle werden zur besseren Übersicht die IEC-Datentypen den Datentypen der Steuerung gegenübergestellt.

Steuerung	zenon	Datenart
BIT, BOOL	BOOL	8
USINT	USINT	9
BYTE	BYTE	22
SINT	SINT	10
WORD	WORD	23
UINT, UINT16	UINT	2
INT, INT16, ENUM	INT	1
UDINT	UDINT	4
DINT	DINT	3
ULINT	ULINT	27
DWORD	DWORD	24
LINT	LINT	26
REAL	REAL	5
LREAL	LREAL	6
STRING	STRING	12
WSTRING	WSTRING	21
DATE	DATE	18
TIME	TIME	17
DT, DATE_AND_TIME	DT	20
TOD, TIME_OF_DAY	TOD (Time of Day)	19

DATENART

Der Begriff **Datenart** ist die interne numerische Bezeichnung des Datentyps. Diese wird auch für den erweiterten DBF Import/Export der Variablen verwendet.

7.4 Variablen anlegen durch Import

Variablen können auch mittels Variablenimport angelegt werden. Für jeden Treiber stehen XML- und DBF-Import zur Verfügung.

> **Info**
>
> Details zu Import und Export von Variablen finden Sie im Handbuch Import-Export (main.chm::/13028.htm) im Abschnitt Variablen (main.chm::/13045.htm).

7.4.1 XML Import

Beim XML- Import von Variablen oder Datentypen werden diese erst einem Treiber zugeordnet und dann analysiert. Vor dem Import entscheidet der Benutzer, ob und wie das jeweilige Element (Variable oder Datentyp) importiert werden soll:

- *Importieren*:
  Das Element wird neu importiert.
- *Überschreiben*:
  Das Element wird importiert und überschreibt ein bereits vorhandenes Element.
- *Nicht importieren*:
  Das Element wird nicht importiert.

Hinweis: Beim Import werden die Aktionen und deren Dauer in einem Fortschrittsbalken angezeigt. In der folgenden Dokumentation wird der Import von Variablen beschrieben. Datentypen werden analog dazu importiert.

VORAUSSETZUNGEN

Beim Import gelten folgende Bedingungen:

- **Abwärtskompatibilität**

  Beim XML Import/Export ist keine Abwärtskompatibilität gegeben. Daten aus älteren zenon Versionen können übernommen werden. Die Übergabe von Daten aus neueren Versionen an ältere wird nicht unterstützt.
- **Konsistenz**

Die zu importierende XML-Datei muss konsistent sein. Beim Import der Datei erfolgt keine Plausibilitätsprüfung. Weisen die importierten Daten Fehler auf, kann es zu unerwünschten Effekten im Projekt kommen.

Dies muss vor allem auch beachtet werden, wenn in einer XML-Datei nicht alle Eigenschaften vorhanden sind und diese dann durch Default-Werte ersetzt werden. Z. B.: Eine binäre Variable hat einen Grenzwert von *300*.

- **Struktur-Datentypen**

  Struktur-Datentypen müssen über die gleiche Anzahl von Strukturelementen verfügen. Beispiel: Ein Strukturdatentyp im Projekt hat 3 Strukturelemente. Ein gleichnamiger Datentyp in der XML-Datei hat 4 Strukturelemente. Dann wird keine der auf diesem Datentyp basierenden Variablen der Datei in das Projekt importiert.

> **Tipp**
>
> Weitere Informationen zum XML-Import finden Sie im Handbuch **Import - Export**, im Kapitel **XML-Import** (**main.chm::/13046.htm**).

7.4.2 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

> **Info**
>
> Import und Export über CSV oder dBase unterstützt keine treiberspezifischen Variableneinstellungen wie z. B. Formeln. Nutzen Sie dafür den Export/Import über XML.

IMPORT DBF-DATEI

Um den Import zu starten:

1. führen Sie einen Rechtsklick auf die Variablenliste aus
2. wählen Sie in der Dropdownliste von **Erweiterter Export/Import** ... den Befehl **dBase importieren**
3. folgen Sie dem Importassistenten

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

Info

Beachten Sie:

- Treiberobjekttyp und Datentyp müssen in der DBF-Datei an den Zieltreiber angepasst werden, damit Variablen importiert werden.
- dBase unterstützt beim Import keine Strukturen oder Arrays (komplexe Variablen).

EXPORT DBF-DATEI

Um den Export zu starten:

1. führen Sie einen Rechtsklick auf die Variablenliste aus
2. wählen Sie im Dropdownliste von **Erweiterter Export/Import** ... den Befehl **dBase exportieren...**
3. folgen Sie dem Exportassistenten

Achtung

DBF-Dateien:

- müssen in der Benennung dem 8.3 DOS Format für Dateinamen entsprechen (8 alphanumerische Zeichen für Name, 3 Zeichen Erweiterung, keine Leerzeichen)
- dürfen im Pfadnamen keinen Punkt (.) enthalten.
  Z. B. ist der Pfad *C:\users\Max.Mustermann\test.dbf* ungültig.
  Gültig wäre: *C:\users\MaxMustermann\test.dbf*
- müssen nahe am Stammverzeichnis (Root) abgelegt werden, um die eventuelle Beschränkungen für Dateinamenlänge inklusive Pfad zu erfüllen: maximal 255 Zeichen

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

Info

dBase unterstützt beim Export keine Strukturen oder Arrays (komplexe Variablen).

DATEIAUFBAU DER DBASE EXPORTDATEI

Für den Variablenimport und -export muss die dBaseIV-Datei folgende Struktur und Inhalte besitzen.

Achtung

dBase unterstützt keine Strukturen oder Arrays (komplexe Variablen).

DBF-Dateien müssen:

- in der Benennung dem 8.3 DOS Format für Dateinamen entsprechen (8 alphanumerische Zeichen für Name, 3 Zeichen Erweiterung, keine Leerzeichen)
- nahe am Stammverzeichnis (Root) abgelegt werden

STRUKTUR

Bezeichnung	Typ	Feldgröße	Bemerkung
KANALNAME	Char	128	Variablenname. Länge kann über den Eintrag **MAX_LAENGE** in der **project.ini** eingeschränkt werden.
KANAL_R	C	128	Ursprünglicher Name einer Variablen, der durch den Eintrag unter VARIABLENNAME ersetzt werden soll (Feld/Spalte muss manuell angelegt werden). Länge kann über den Eintrag **MAX_LAENGE** in der **project.ini** eingeschränkt werden.
KANAL_D	Log	1	Variable wird bei Eintrag *1* gelöscht (Feld/Spalte muss manuell angelegt werden).
TAGNR	C	128	Kennung. Länge kann über den Eintrag **MAX_LAENGE** in der **project.ini** eingeschränkt werden.
EINHEIT	C	11	Technische Maßeinheit
DATENART	C	3	Datentyp (z. B. Bit, Byte, Wort, ...) entspricht dem Datentyp.
KANALTYP	C	3	Speicherbereich in der SPS (z. B. Merkerbereich, Datenbereich, ...) entspricht Treiberobjekttyp.
HWKANAL	Num	3	Netzadresse
BAUSTEIN	N	3	Datenbaustein-Adresse (nur bei Variablen aus den Datenbereich der SPS)

Bezeichnung	Typ	Feldgröße	Bemerkung
ADRESSE	N	5	Offset
BITADR	N	2	Für Bit-Variablen: Bitadresse Für Byte-Variablen: 0=niederwertig, 8=höherwertig Für String-Variablen: Stringlänge (max. 63 Zeichen)
ARRAYSIZE	N	16	Anzahl der Variablen im Array für Index-Variablen ACHTUNG: Nur die erste Variable steht voll zur Verfügung. Alle folgenden sind nur über VBA oder den Rezeptgruppen Manager zugänglich
LES_SCHR	L	1	Lese-Schreib-Berechtigung 0: Sollwert setzen ist nicht erlaubt 1: Sollwert setzen ist erlaubt
MIT_ZEIT	L	1	Zeitstempelung in zenon (nur wenn vom Treiber unterstützt)
OBJEKT	N	2	Treiberspezifische ID-Nummer des Primitivobjekts setzt sich zusammen aus TREIBER-OBJEKTTYP und DATENTYP
SIGMIN	Floa t	16	Rohwertsignal minimal (Signalauflösung)
SIGMAX	F	16	Rohwertsignal maximal (Signalauflösung)
ANZMIN	F	16	technischer Wert minimal (Messbereich)
ANZMAX	F	16	technischer Wert maximal (Messbereich)
ANZKOMMA	N	1	Anzahl der Nachkommastellen für die Darstellung der Werte (Messbereich)
UPDATERATE	F	19	Updaterate für Mathematikvariablen (in sec, eine Dezimalstelle möglich) bei allen anderen Variablen nicht verwendet
MEMTIEFE	N	7	Nur aus Kompatibilitätsgründen vorhanden
HDRATE	F	19	HD-Updaterate für hist. Werte (in sec, eine Dezimalstelle möglich)
HDTIEFE	N	7	HD-Eintragtiefe für hist. Werte (Anzahl)
NACHSORT	L	1	HD-Werte als nachsortierte Werte
DRRATE	F	19	Aktualisierung an die Ausgabe (für zenon DDE-Server,

Bezeichnung	Typ	Feldgröße	Bemerkung
			in sec, eine Kommastelle möglich)
HYST_PLUS	F	16	Positive Hysterese; ausgehend vom Messbereich
HYST_MINUS	F	16	Negative Hyterese; ausgehend vom Messbereich
PRIOR	N	16	Priorität der Variable
REAMATRIZE	C	32	Name der zugeordnete Reaktionsmatrix
ERSATZWERT	F	16	Ersatzwert; ausgehend vom Messbereich
SOLLMIN	F	16	Sollwertgrenze Minimum; ausgehend vom Messbereich
SOLLMAX	F	16	Sollwertgrenze Maximum; ausgehend vom Messbereich
VOMSTANDBY	L	1	Variable vom Standby Server anfordern; der Wert der Variable wird im redundanten Netzwerkbetrieb nicht vom Server sondern vom Standby Server angefordert
RESOURCE	C	128	Betriebsmittelkennung. Freier String für Export und Anzeige in Listen. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
ADJWVBA	L	1	Nichtlineare Wertanpassung: *0*: Nichtlineare Wertanpassung wird verwendet *1*: Nichtlineare Wertanpassung wird nicht verwendet
ADJZENON	C	128	Verknüpftes VBA-Makro zum Lesen der Variablenwerte für die nichtlineare Wertanpassung.
ADJWVBA	C	128	Verknüpftes VBA-Makro zum Schreiben der Variablenwerte für die nichtlineare Wertanpassung.
ZWREMA	N	16	Verknüpfte Zählwert-Rema.
MAXGRAD	N	16	Maximaler Gradient für die Zählwert-Rema.

Achtung

Beim Import müssen Treiberobjekttyp und Datentyp in der DBF-Datei an den Zieltreiber angepasst werden, damit Variablen importiert werden.

GRENZWERTDEFINITION

Grenzwertdefinition für Grenzwert *1* bis *4*, oder Zustand *1* bis *4*:

Bezeichnung	Typ	Feldgröße	Bemerkung
AKTIV1	L	1	Grenzwert aktiv (pro Grenzwert vorhanden)
GRENZWERT1	F	20	technischer Wert oder ID-Nummer der verknüpften Variable für einen dynamischen Grenzwert (siehe VARIABLEx) (wenn unter VARIABLEx *1* steht und hier *-1*, wird die bestehende Variablenzuordnung nicht überschrieben)
SCHWWERT1	F	16	Schwellwert für den Grenzwert
HYSTERESE1	F	14	wird nicht verwendet
BLINKEN1	L	1	Blinkattribut setzen
BTB1	L	1	Protokollierung in CEL
ALARM1	L	1	Alarm
DRUCKEN1	L	1	Druckerausgabe (bei CEL oder Alarm)
QUITTIER1	L	1	quittierpflichtig
LOESCHE1	L	1	löschpflichtig
VARIABLE1	L	1	dyn. Grenzwertverknüpfung der Grenzwert wird nicht durch einen absoluten Wert (siehe Feld GRENZWERTx) festgelegt.
FUNC1	L	1	Funktionsverknüpfung
ASK_FUNC1	L	1	Ausführung über die Alarmmeldeliste
FUNC_NR1	N	10	ID-Nummer der verknüpften Funktion (steht hier -1, so wird die bestehende Funktion beim Import nicht überschrieben)
A_GRUPPE1	N	10	Alarm/Ereignis-Gruppe
A_KLASSE1	N	10	Alarm/Ereignis-Klasse
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Farbe als Windowskodierung

Bezeichnung	Typ	Feldgröße	Bemerkung
GRENZTXT1	C	66	Grenzwerttext
A_DELAY1	N	10	Zeitverzögerung
INVISIBLE1	L	1	Unsichtbar

Bezeichnungen in der Spalte Bemerkung beziehen sich auf die in den Dialogboxen zur Definition von Variablen verwendeten Begriffe. Bei Unklarheiten, siehe Kapitel Variablendefinition.

7.4.3 Online- und Offline-Import

Der Import von Variablen aus symbolischen Variablennamen kann wahlweise Online aus der Steuerung oder offline über eine Symboldatei erfolgen.

VORAUSSETZUNGEN FÜR DEN IMPORT

Damit der Variablenimport funktioniert, müssen die im zenon Projekt verwendeten Variablen in der **CoDeSys** Projektiersoftware über ein Symbolkonfigurations-Objekt ausgewählt, und in die Steuerung übertragen werden. Bei diesem Vorgang wird gleichzeitig eine Symboldatei im XML-Format erzeugt, die für den Offline-Variablenimport verwendet werden kann.

Je nach den Einstellungen, ob die **Variablen-Kennung** oder die **Symbolische Adresse** für die Adressierung verwendet werden soll, wird der **Symbolische Name** in der entsprechenden Eigenschaft eingetragen.

VARIABLENNAME

Als Variablenname wird der importierte **Symbolische Name** mit eine Präfix eingetragen.

Dieses Präfix ist:

- Alias der konfigurierten Verbindung
- Knotenadresse der konfigurierten Verbindung, wenn kein Alias konfiguriert ist.

Hinweis: Diese Konfiguration erstellen Sie im Treiberdialog beim Anlegen einer neuen Verbindung (auf Seite 24).

Damit ist gewährleistet, dass der Variablenname auch auch bei gleichlautenden Symbolnamen in verschiedenen Steuerungen eindeutig bleibt. Der Variablenname kann daher nicht für die symbolische Adressierung verwendet werden.

DATENTYP ENUM - IMPORT AUS XML-SYMBOLDATEI

Für den Import von ENUM-Datentypen aus einer XML-Symboldatei gilt:

- Ist für einem Datentyp *typeclass=ENUM* definiert und ist dafür ein *basetype* angegeben, so wird die Variable in zenon mit diesem *basetype* angelegt.
- Ist kein *basetype* angegeben, wird für die Variable in zenon der Datentyp INT verwendet.

7.4.3.1 Variablenimport mit Wizard

Um Variablen direkt von der SPS zu importieren:

1. Wählen Sie im Kontextmenü des Treibers **Variablen vom Treiber importieren**.
   Die Import-Wizard wird geöffnet.
2. Wählen Sie im Dialog die Importquelle:
   - Steuerung:
     Die Variablen werden direkt von der SPS importiert.
   - XML-Symboldatei
     Die Variablen werden von einer XML-Datei importiert.
3. Klicken Sie auf **Weiter.**
4. Wählen Sie die Steuerung und bei Offline-Import auch die XML-Datei aus.
   Die Liste der Steuerungen entspricht der Konfiguration der Treiberkonfiguration (wie in Verbindung (auf Seite 24) konfiguriert).
5. Klicken Sie auf **Weiter.**
6. Konfigurieren Sie die Filtereinstellungen.
   Weitere Informationen dazu finden Sie in der Beschreibung des Filterdialogs.
7. Klicken Sie auf **Fertig.**
   Die Auswahlliste wird angezeigt.
8. Wählen Sie die zu importierenden Variablen aus.
9. Klicken Sie auf **OK**.

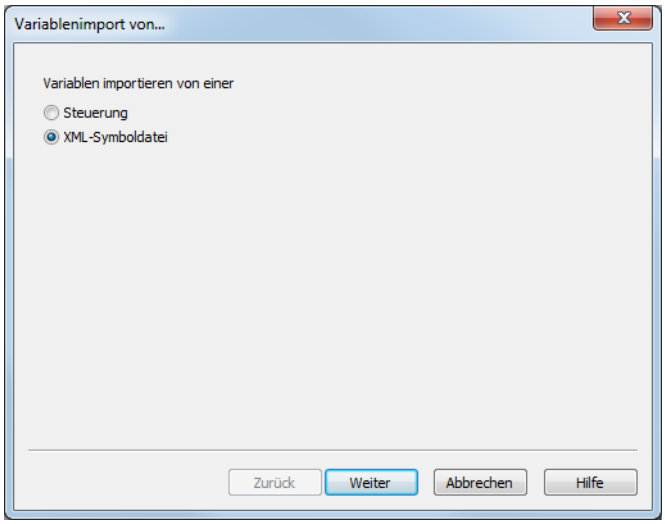
Die Variablen werden importiert. Sollte ein Fehler auftreten, wird eine entsprechende Meldung ausgegeben.

7.4.3.2 Online- und Offline-Import

Während des Importvorganges wählen Sie mit dem Import-Wizard die gewünschten Optionen.

AUSWAHL DER IMPORTQUELLE

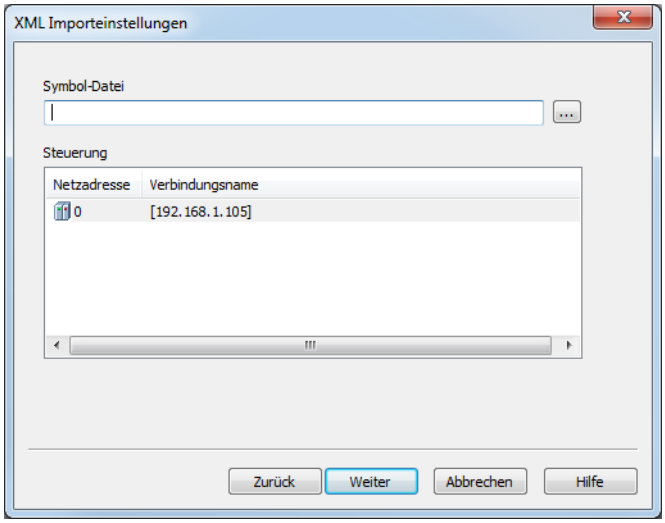
In diesem Dialog wählen Sie die Importquelle für den Variablenimport.



Parameter	Beschreibung
Variablen importieren aus	Optionsfeld zur Auswahl der Importquelle. ▸ *Steuerung*: Variablen werden aus einer Steuerung importiert. ▸ *XML-Symboldatei*: Variablen werden aus einer mit der CoDeSys-Software erstellten XML-Datei importiert.
Weiter	Übernimmt Einstellung und wechselt zum nächsten Fenster.
Abbrechen	Bricht den Import ab. Es werden keine Variablen importiert.
Hilfe	Öffnet die Online-Hilfe.

QUELLE XML-DATEI: AUSWAHL DER XML-DATEI UND DER STEUERUNG

In diesem Dialog wählen Sie die Quelle der zu importierenden Variablen. Bei Import aus einer XML-Symboldatei wird zusätzlich die Datei angegeben.

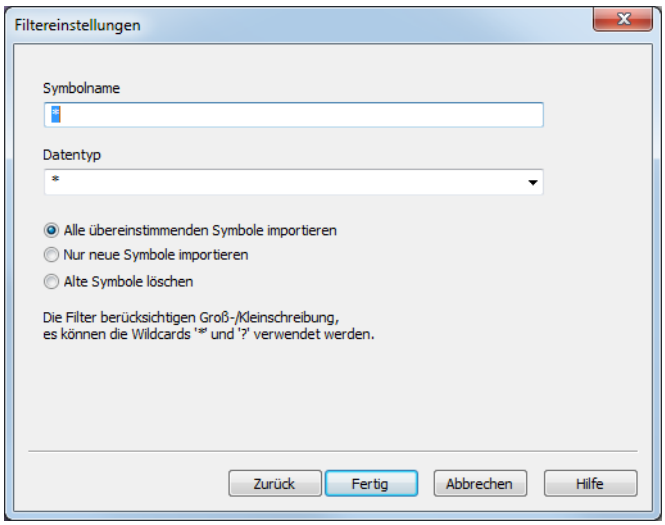


Parameter	Beschreibung
Symbol-Datei	Auswahl der Symboldatei. Nur vorhanden, wenn bei der Auswahl der Importquelle **XML-Symboldatei** als Quelle ausgewählt wurde.
Steuerung	Auswahl der Steuerung, deren Variablen importiert werden, aus Liste. Die Liste der Steuerungen entspricht der Konfiguration der Treiberkonfiguration (wie in Verbindung (auf Seite 24) konfiguriert).
Zurück	Wechselt zum vorangegangenen Fenster.
Weiter	Übernimmt Einstellung und wechselt zum nächsten Fenster. **Hinweis:** Ist keine Verbindung konfiguriert oder ausgewählt, kann nicht zum nächsten Fenster gewechselt werden.
Abbrechen	Bricht den Import ab. Es werden keine Variablen importiert.
Hilfe	Öffnet die Online-Hilfe.

KONFIGURATION DES FILTERS

Beim Variablenimport können die zu importierenden Variablen gefiltert werden.

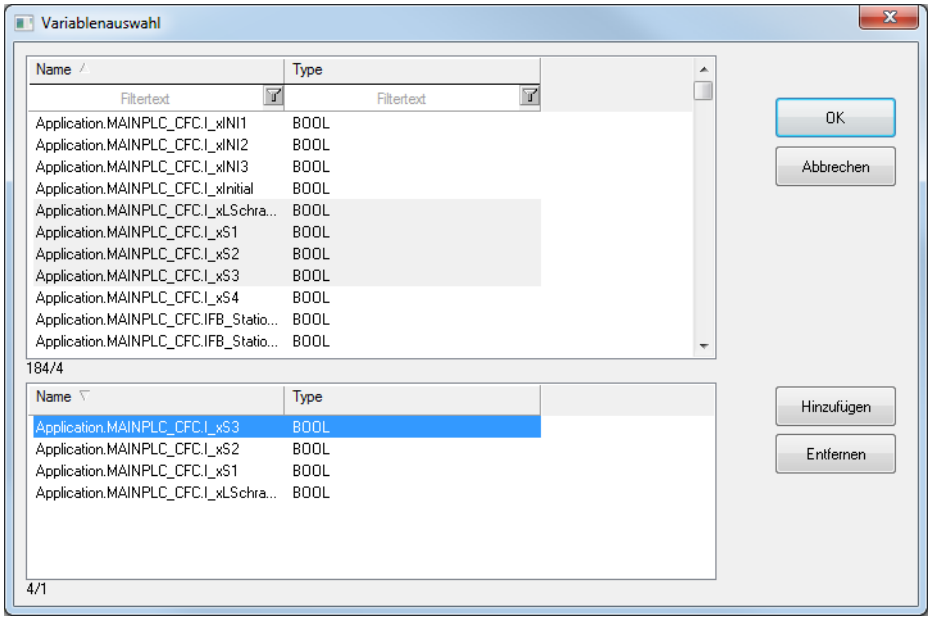
Es werden entweder alle vorhandenen oder nur neue Variablen zur Auswahl angeboten. Zusätzlich könne nicht mehr benötigte Variablen gesucht und gelöscht werden.



Parameter	Beschreibung
Symbolname	Eingabe eines Symbolnamens als Filterkriterium. Der Name kann mit folgenden Platzhaltern eingegeben werden: ▸ ***** ▸ **?** Default: * Wählt alle Symbolnamen aus, keine Vorfilterung.
Datentyp	Eingabe eines Datentyps in Combobox oder Auswahl aus Dropdownliste. Bei der Eingabe können folgende Platzhalter verwendet werden: ▸ ***** ▸ **?** Default: * Wählt alle Datentypen aus, keine Vorfilterung.
Vorfilterung	Optionsfeld zur Vorfilterung der angezeigten Variablen im Folgedialog (Variablenauswahldialog): ▸ *Alle übereinstimmenden Symbole importieren* Bietet alle mit dem Filter übereinstimmenden Symbole zum Import an. ▸ *Nur neue Symbole importieren* Bietet nur neue, im aktuellen zenon Projekt noch nicht

Parameter	Beschreibung
	vorhandene, Symbole zum Import an. ▸ *Alte Symbole löschen* Bietet alle im aktuellen zenon Projekt vorhandenen Symbole, die auf der Steuerung nicht existieren, zum Löschen an.
Zurück	Wechselt zum vorangegangenen Fenster.
Fertig	Übernimmt Einstellung, und öffnet den Variablenauswahldialog.
Abbrechen	Bricht den Import ab. Es werden keine Variablen importiert.
Hilfe	Öffnet die Online-Hilfe.

VARIABLENAUSWAHLDIALOG



Parameter	Beschreibung
Liste importierbare Variablen	Zeigt alle Variablen an, die entsprechend den Einstellungen im Wizard importiert oder gelöscht werden können. ▸ *Name* Name der Variable wie in der Steuerung oder in der XML-Symboldatei definiert. ▸ *Type* Typ der Variable wie in der Steuerung oder in der XML-Symboldatei definiert.

Parameter	Beschreibung
	▸ *Access* Die Liste ist sortier- und filterbar.
Liste zu importierende Variablen	Zeigt alle Variablen an, die für den Import oder zum Löschen ausgewählt wurden.
Hinzufügen	Fügt die in der Liste **Importierbare Variablen** markierten Variablen zur Liste **Zu importierende Variablen** hinzu. Mehrfachauswahl aus der Liste ist möglich.
Entfernen	Löscht markierte Variablen aus Liste **Zu importierende Variablen**. Mehrfachauswahl aus der Liste ist möglich.
OK	Übernimmt Auswahl und importiert oder löscht Variablen.
Abbrechen	Bricht den Import ab und schließt den Wizard. Es werden keine Variablen importiert.

7.5 Kommunikationsdetails (Treibervariablen)

Das Treiberkit implementiert eine Reihe von Treibervariablen, welche in dem Treiberobjekttyp *Kommunikationsdetails* zusammengefasst sind. Diese sind unterteilt in:

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

Die Definitionen der im Treiberkit implementierten Variablen sind in der Importdatei **DRVVAR.DBF** verfügbar und können von dort importiert werden.
Pfad zur Datei: *%ProgramData%\COPA-DATA\zenon<Versionsnummer>\PredefinedVariables*

Hinweis: Variablennamen müssen in zenon einzigartig sein. Soll nach einem Import der Variablen vom Treiberobjekttyp *Kommunikationsdetails* aus **DRVVAR.DBF** ein erneuter Import durchgeführt werden, müssen die zuvor importierten Variablen umbenannt werden.

Info

Nicht jeder Treiber unterstützt alle Treibervariablen des Treiberobjekttyps *Kommunikationsdetails*.

Zum Beispiel:

- Variablen für Modem-Informationen werden nur von modemfähigen Treibern unterstützt.
- Treibervariablen für den Polling-Zyklus stehen nur für rein pollende Treiber zur Verfügung.
- Verbindungsbezogene Informationen wie **ErrorMSG** werden nur von Treibern unterstützt, die zu einem Zeitpunkt nur eine Verbindung bearbeiten.

INFORMATION

Name aus Import	Typ	Offset	Erklärung
MainVersion	*UINT*	0	Haupt-Versionsnummer des Treibers.
SubVersion	*UINT*	1	Sub-Versionsnummer des Treibers.
BuildVersion	*UINT*	29	Build-Versionsnummer des Treibers.
RTMajor	*UINT*	49	zenon Hauptversionsnummer
RTMinor	*UINT*	50	zenon Sub-Versionsnummer
RTSp	*UINT*	51	zenon Service Pack-Nummer
RTBuild	*UINT*	52	zenon Buildnummer
LineStateIdle	*BOOL*	24.0	TRUE, wenn die Modemleitung belegt ist.
LineStateOffering	*BOOL*	24.1	TRUE, wenn ein Anruf rein kommt.
LineStateAccepted	*BOOL*	24.2	Der Anruf wird angenommen.
LineStateDialtone	*BOOL*	24.3	Rufton wurde erkannt.
LineStateDialing	*BOOL*	24.4	Wahl aktiv.
LineStateRingBack	*BOOL*	24.5	Während Verbindungsaufbau.
LineStateBusy	*BOOL*	24.6	Zielstation besetzt.
LineStateSpecialInfo	*BOOL*	24.7	Spezielle Statusinformation empfangen.
LineStateConnected	*BOOL*	24.8	Verbindung hergestellt.
LineStateProceeding	*BOOL*	24.9	Wahl ausgeführt.
LineStateOnHold	*BOOL*	24.10	Verbindung in Halten.

Name aus Import	Typ	Offset	Erklärung
LineStateConferenced	*BOOL*	24.11	Verbindung im Konferenzmodus.
LineStateOnHoldPendConf	*BOOL*	24.12	Verbindung in Halten für Konferenz.
LineStateOnHoldPendTransfer	*BOOL*	24.13	Verbindung in Halten für Transfer.
LineStateDisconnected	*BOOL*	24.14	Verbindung beendet.
LineStateUnknow	*BOOL*	24.15	Verbindungszustand nicht bekannt.
ModemStatus	*UDINT*	24	Aktueller Modemstatus.
TreiberStop	*BOOL*	28	Treiber gestoppt Bei *Treiberstop*, hat die Variable den Wert *TRUE* und ein **OFF**-Bit. Nach dem Treiberstart, hat die Variable den Wert *FALSE* und kein **OFF**-Bit.
SimulRTState	*UDINT*	60	Informiert über Status der Runtime bei Treibersimulation.
ConnectionStates	*STRING*	61	Interner Verbindungsstatus des Treibers zur SPS. Verbindungszustände: *0:* Verbindung OK *1:* Verbindung gestört *2:* Verbindung simuliert Formatierung: *<Netzadresse>:<Verbindungszustand>;...;...;* Eine Verbindung ist erst nach dem ersten Anmelden einer Variablen bekannt. Damit eine Verbindung im String enthalten ist, muss einmal eine Variable dieser Verbindung angemeldet worden sein. Der Zustand einer Verbindung wird nur aktualisiert, wenn eine Variable der Verbindung angemeldet ist. Ansonsten wird nicht mit der entsprechenden Steuerung

Name aus Import	Typ	Offset	Erklärung
			kommuniziert.

KONFIGURATION

Name aus Import	Typ	Offset	Erklärung
ReconnectInRead	*BOOL*	27	Wenn TRUE, dann wird beim Lesen automatisch ein Neuaufbau der Verbindung durchgeführt.
ApplyCom	*BOOL*	36	Änderungen an den Einstellungen der seriellen Schnittstelle zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyCom zur Folge (aktuell ohne weitere Funktion).
ApplyModem	*BOOL*	37	Änderungen an den Modemeinstellungen zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyModem zur Folge. Diese schließt die aktuelle Verbindung und öffnet eine neue entsprechend den Einstellungen **PhoneNumberSet** und **ModemHwAdrSet**.
PhoneNumberSet	*STRING*	38	Telefonnummer, welche verwendet werden soll.
ModemHwAdrSet	*DINT*	39	Hardwareadresse, welche zu der Telefonnummer gehört.
GlobalUpdate	*UDINT*	3	Updatezeit in Millisekunden (ms).
BGlobalUpdaten	*BOOL*	4	TRUE, wenn die Updatezeit global ist.
TreiberSimul	*BOOL*	5	TRUE, wenn der Treiber in Simulation ist.
TreiberProzab	*BOOL*	6	TRUE, wenn das Prozessabbild gehalten werden soll.
ModemActive	*BOOL*	7	TRUE, wenn das Modem bei diesem Treiber aktiv ist.
Device	*STRING*	8	Name der seriellen Schnittstelle oder Name des Modem.
ComPort	*UINT*	9	Nummer der seriellen Schnittstelle.

Name aus Import	Typ	Offset	Erklärung
Baudrate	*UDINT*	10	Baudrate der seriellen Schnittstelle.
Parity	*SINT*	11	Parität der seriellen Schnittstelle.
ByteSize	*USINT*	14	Bitanzahl pro Zeichen der seriellen Schnittstelle. Wert = *0*, wenn der Treiber keine serielle Kommunikation herstellen kann.
StopBit	*USINT*	13	Anzahl der Stoppbits der seriellen Schnittstelle.
Autoconnect	*BOOL*	16	TRUE, wenn die Modemverbindung automatisch beim Lesen/Schreiben aufgebaut werden soll.
PhoneNumber	*STRING*	17	Aktuelle Telefonnummer.
ModemHwAdr	*DINT*	21	Hardwareadresse zur aktuellen Telefonnummer.
RxIdleTime	*UINT*	18	Wenn länger als diese Zeit in Sekunden (s) erfolgreich kein Datenverkehr stattfindet, wird die Modemverbindung beendet.
WriteTimeout	*UDINT*	19	Maximale Schreibdauer bei einer Modemverbindung in Millisekunden (ms).
RingCountSet	*UDINT*	20	So oft läutet ein hereinkommender Anruf, bevor dieser angenommen wird.
ReCallIdleTime	*UINT*	53	Wartezeit zwischen Anrufen in Sekunden (s).
ConnectTimeout	*UINT*	54	Zeit in Sekunden (s) für Verbindungsaufbau.

STATISTIK

Name aus Import	Typ	Offset	Erklärung
MaxWriteTime	*UDINT*	31	Längste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MinWriteTime	*UDINT*	32	Kürzeste Zeit in Millisekunden (ms), die zum

Name aus Import	Typ	Offset	Erklärung
			Schreiben benötigt wird.
MaxBlkReadTime	*UDINT*	40	Längste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
MinBlkReadTime	*UDINT*	41	Kürzeste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
WriteErrorCount	*UDINT*	33	Anzahl der Schreibfehler.
ReadSucceedCount	*UDINT*	35	Anzahl der erfolgreichen Leseversuche.
MaxCycleTime	*UDINT*	22	Längste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
MinCycleTime	*UDINT*	23	Kürzeste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
WriteCount	*UDINT*	26	Anzahl der Schreibversuche.
ReadErrorCount	*UDINT*	34	Anzahl der fehlerhaften Leseversuche.
MaxUpdateTimeNormal	*UDINT*	56	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Normal** in Millisekunden (ms).
MaxUpdateTimeHigher	*UDINT*	57	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höher** in Millisekunden (ms).
MaxUpdateTimeHigh	*UDINT*	58	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Hoch** in Millisekunden (ms).
MaxUpdateTimeHighest	*UDINT*	59	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höchste** in Millisekunden (ms).
PokeFinish	*BOOL*	55	Geht für eine Abfrage auf *1*, wenn alle anstehenden Pokes ausgeführt wurden.

FEHLERMELDUNGEN

Name aus Import	Typ	Offset	Erklärung
ErrorTimeDW	*UDINT*	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler auftrat.
ErrorTimeS	*STRING*	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler als String auftrat.

Name aus Import	Typ	Offset	Erklärung
RdErrPrimObj	*UDINT*	42	Nummer des PrimObjektes, als der letzte Lesefehler verursacht wurde.
RdErrStationsName	*STRING*	43	Name der Station, als der letzte Lesefehler verursacht wurde.
RdErrBlockCount	*UINT*	44	Anzahl der zu lesenden Blöcke, als der letzte Lesefehler verursacht wurde.
RdErrHwAdresse	*DINT*	45	Hardwareadresse, als der letzte Lesefehler verursacht wurde.
RdErrDatablockNo	*UDINT*	46	Bausteinnummer, als der letzte Lesefehler verursacht wurde.
RdErrMarkerNo	*UDINT*	47	Merkernummer, als der letzte Lesefehler verursacht wurde.
RdErrSize	*UDINT*	48	Blockgröße, als der letzte Lesefehler verursacht wurde.
DrvError	*USINT*	25	Fehlermeldung als Nummer.
DrvErrorMsg	*STRING*	30	Fehlermeldung als Klartext.
ErrorFile	*STRING*	15	Name der Fehlerprotokolldatei.

8 Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

8.1 Redundanz

Für den 3S_V3 Treiber sind folgende Arten der Kommunikation möglich:

- Nur primäre Verbindung konfiguriert
  Ist nur die primäre Verbindung konfiguriert, erfolgt die Kommunikation mit der Steuerung über eine einzelne Verbindung. Fällt diese aus, ist keine Kommunikation möglich.
  Konfiguration im Treiberdialog: Option **Primäre Knotenadresse**

- Primäre und sekundäre Verbindung konfiguriert:
  Der Treiber verbindet sich per Default mit der primären Verbindung. Kann diese Verbindung nicht hergestellt werden, erfolgt der Verbindungsaufbau mit der sekundären Verbindung. Fällt die primäre Verbindung zu einer SPS aus, erfolgt die Kommunikation mit der sekundären Verbindung.
  Die primäre Verbindung hat immer Vorrang. Sind beide konfigurierten Verbindungen verfügbar, wird immer die primäre Verbindung verwendet.
  Konfiguration im Treiberdialog: Optionen **Primäre Knotenadresse** und **Sekundäre Knotenadresse**
  Dabei werden alle Variablen immer nur bei der aktiven Verbindung angemeldet (*advised*). Dies bedeutet, dass Werte immer nur von der aktiven Verbindung gelesen werden. Das Schreiben von Werten erfolgt immer auf beide Steuerungen.
- Dabei ist zu beachten, dass der Wert der Variable immer von der aktiven Verbindung gelesen wird. Beiden Steuerungen müssen daher extern synchronisiert sein und eine entsprechende Logik implementiert haben.
- Variable sowie primäre und sekundäre Verbindung konfiguriert:
  Sind beide Verbindungen konfiguriert und verfügbar können alle Verbindungsstatus mit einer Variable vom Treiberobjekttyp **Verbindungsstatus** ausgelesen werden:

TREIBEROBJEKTTYP VERBINDUNGSSTATUS

Zusammensetzung des Datentyps *USINT*:

Bit Position	Bitwert 0	Bitwert 1
0	Primäre Verbindung fehlt	Primäre Verbindung aufgebaut
1	Sekundäre Verbindung fehlt	Sekundäre Verbindung aufgebaut
2	Primäre Verbindung inaktiv	Primäre Verbindung aktiv
3	Sekundäre Verbindung inaktiv	Sekundäre Verbindung aktiv

Mit dem Datentyp *BOOL* kann durch Angabe der entsprechenden Bitnummer das jeweilige Statusbit direkt ausgelesen werden.

Achtung

Bei gleichzeitiger Verwendung von Redundanz und Exception Handling werden die Variablen vom Treiberobjekttyp **Status Feedback** zur Rückmeldung des Anlauf-Status an die SPS immer parallel an beide Steuerungen geschrieben.

Die Alarme werden allerdings nur von der aktiven Verbindung gelesen.

Stellen Sie daher sicher, dass die Speicherblöcke für das Exception Handling in der Steuerung so synchronisiert werden, dass beim Wechsel der Verbindung keine Alarme verloren gehen oder doppelt gemeldet werden.

Durch die (optionale) Angabe einer alternativen (sekundären) Knotenadresse ist der Aufbau einer Verbindung zu einer redundanten Steuerung möglich. Zusätzlich kann durch eine boolsche Variable die aktive Verbindung gewählt werden (ebenfalls optional)

TREIBEROBJEKTTYP VERBINDUNGSSTATUS

Um den Status der jeweiligen Verbindung anzuzeigen, wird eine Variable vom Treiberobjekttyp **Verbindungsstatus** verwendet.

Für den Datentyp USINT setzt er sich wie folgt zusammen:

Bit	Bitwert 0	Bitwert 1
0	Primäre Verbindung fehlt	Primäre Verbindung aufgebaut
1	Sekundäre Verbindung fehlt	Sekundäre Verbindung aufgebaut

Bit	Bitwert 0	Bitwert 1
2	Primäre Verbindung inaktiv	Primäre Verbindung aktiv
3	Sekundäre Verbindung inaktiv	Sekundäre Verbindung aktiv

Mit dem Datentyp BOOL kann durch Angabe der entsprechenden Bitnummer das jeweilige Statusbit direkt ausgelesen werden

8.2 Rückmeldung zur SPS

Während der Herstellung einer Verbindung sollte die SPS verhindern, dass sich Werte im Ereignis-Array ändern.

Dazu legen Sie in zenon für jede Verbindung zu einer SPS eine Variable mit dem Treiberobjekttyp **Status Feedback** an. Diese meldet den aktuellen Anlauf-Status an die SPS. Diese Variable kann folgende Werte annehmen:

- *-1*: *Not Running*
  Verbindung beendet.
  Vor dem ordnungsgemäßen Beenden der Verbindung (beim Beenden der Runtime) wird die Status-Feedback-Variable auf den Wert Not Running gesetzt.
- *0*: *Initial*
  Verbindung wiederhergestellt, Beginn des Wiederanlaufs.
  Der Wert tritt nur beim Wiederherstellen der Verbindung auf, nicht beim Starten der Runtime. Zu diesem Zeitpunkt sind noch keine Variablen von der Runtime angefordert, daher kann auch deren Wert nicht gesetzt werden
- *10*: *Starting*
  Während der Anforderung von Variablen bei der SPS.
- *100*: *Running*
  Nachdem die Anfangswerte von der SPS empfangen wurden.

Die SPS soll das Eintragen neuer Ereignisse verhindern, während die *Status Feedback* Variable den Wert *Starting* hat.

8.3 Byte-Reihenfolge

Der **3S V3** Treiber verarbeitet Daten mit der Intel Byte-Reihenfolge (*Byte Order*). Daten, die von der Steuerung in Motorola Byte-Reihenfolge übermittelt werden, werden vom Treiber angepasst, wenn die Option **Motorola Byte Order** im Konfigurationsdialog der Verbindung (auf Seite 24) aktiviert ist.

Zusätzlich kann die Byte-Reihenfolge anhand des verwendeten RDA-Typs automatisch ermittelt werden.

Der Treiber erkennt automatisch aufgrund des RDA-Typs in welcher Byte-Reihenfolge die Daten von der Steuerung übermittelt werden und wandelt diese Intern für die Verarbeitung der Daten in zenon an. Bei der Kommunikation zur SPS findet ebenfalls eine entsprechende Anpassung der Byte-Reihenfolge statt.

ERKENNUNG DER BYTE-REIHENFOLGE AUF BASIS DES RDA-TYPS

Für die automatische Erkennung der Byte-Reihenfolge durch den **3S V3** Treiber gilt:

- Intel Byte-Reihenfolge

  RDA-Typ: *1*, *2*, *3* oder *4*

  Header und Daten liegen im Intel-Format vor. Daten werden direkt durch den **3S V3** Treiber verarbeitet. Dabei ist keine zusätzliche Anpassung durch den Treiber notwendig.

- Motorola Byte-Reihenfolge

  Header und Daten liegen im Motorola-Format vor. Zusätzlich ist die Option **Motorola Byte Order** im Konfigurationsdialog der Verbindung aktiviert.
  RDA-Typ: *0x1000000*, *0x2000000*, *0x3000000* oder *0x4000000*

  Header und Daten liegen im Motorola-Format vor und werden vom Treiber intern angepasst.

- Motorola Byte-Reihenfolge – Byte-Reihenfolge durch SPS angepasst

  Die Option **Motorola Byte Order** ist im Konfiugrationsdialog der Verbindung aktiviert. Zusätzlich wird die Byte-Reihenfolge bereits durch die SPS an die Intel Byte-Reihenfolge angepasst.
  RDA-Typ: *1*, *2*, *3* oder *4*

  Header und Daten werden im Intel-Format von der SPS kommuniziert. Daten werden direkt durch den **3S V3** Treiber verarbeitet. Dabei ist keine zusätzliche Anpassung durch den Treiber notwendig.

BYTE-REIHENFOLGE - ÜBERSICHT

Intel	Motorola
1	*0x1000000*
2	*0x2000000*
3	*0x3000000*
4	*0x4000000*

Intel **Motorola**

Achtung

Für den 3S V3 RDA-Mechanismus bestimmt die Größe des Triggers in der Steuerung auch die Größe des Datenarrays. Für eine einwandfreie Kommunikation müssen beide Größen gleich sein. Die vier Header-Felder sind immer 4 Byte groß. Manche Steuerungen erlauben aber keine gemischte Adressierung. Dadurch kann es zu zu Problemen kommen, wenn die Triggervariable kleiner als 4 Byte ist. In diesem Fall werden Header und Daten falsch ausgelesen.

8.4 Exception Handling und Alarmierung

Mit Hilfe des Exception-Handlings können Fehlerereignisse von der Steuerung an zenon gemeldet und über eine ID einer Alarmvariablen zugeordnet werden.

Für den Austausch von Fehlerereignissen werden in der Steuerung mehrere Blöcke der folgenden Form - z.B. in einem Array - angelegt:

```
TYPE ST_ISA_ALARM :
STRUCT
    ID : DINT;                      // Kennung für Fehlerort und -Art
    Value : DINT;                   // Fehlerwert, Code, ...
    Message : STRING(80);           // Fehlermeldung
    TimeEvent.AlmDate : DATE;       // Datum Kommt (UTC)
    TimeEvent.AlmTime : TIME_OF_DAY; // Zeit Kommt (UTC)
    TimeAck.AlmDate : DATE;         // Datum Geht (UTC)
    TimeAck.AlmTime : TIME_OF_DAY;  // Zeit Geht (UTC)
    AlarmState : USINT;             // Alarmstatus (Kommt/Geht)
    ComState : INT;                 // Handshake
END_STRUCT
END_TYPE
```

Dabei gilt:

- Die Namen der Strukturfelder können frei vergeben werden.

- Der Aufbau der Struktur (Datentypen) muss beibehalten werden.
  Anderenfalls können die Daten eines Ereignisses nur fehlerhaft oder unvollständig gelesen werden.

Ablauf:

- Beim Auftreten eines Fehlerereignisses belegt die Steuerung einen beliebigen freien Block (Handshake-Variable = *2*) mit den Angaben zum Ereignis und setzt anschließend dessen Handshake-Variable auf *1*. Das signalisiert dem Treiber, dass der betreffende Block neue Daten enthält.
- Beim Wechsel der Handshake-Variable auf *1* liest der Treiber den gesamten Block ein und weist den Inhalt der über die ID festgelegten Alarmvariablen in zenon zu.
- Die Symboladresse des Blocks wird aus der Symboladresse der Handshake-Variablen durch Abschneiden des Teils nach dem letzten Punkt ermittelt.
  Zum Beispiel: Symboladresse der Handshake-Variable ist **Alarm[8].ComState**. Das ergibt die Symboladresse des Blocks **Alarm[8]**.
- Anschließend setzt der Treiber die Handshake-Variable auf *2*. Das signalisiert der Steuerung, dass der Block wieder frei ist und für neue Ereignisse verwendet werden kann.
- Tritt während der Auswertung ein Fehler auf, setzt der Treiber die Handshake-Variable auf den Wert *4* und signalisiert damit der Steuerung den Fehlerzustand.
  Fehler sind zum Beispiel: keine Alarmvariable mit der ID verknüpft, Fehler beim Lesen, etc.

ALARMSTATUS

Der Inhalt des Feldes AlarmState bestimmt, ob der Alarm kommt oder geht. Abhängig davon werden der Wert und der Zeitstempel (in UTC) der verbundenen Alarmvariablen ermittelt:

- *0*: Leer.
  Kein Alarm vorhanden, alle anderen Einträge des Blocks sind ungültig.
- *1*: Alarm kommt.
  Die Alarmvariable erhält den Wert *1* und den Zeitstempel aus TimeEvent.AmlDate und TimeEvent.AmlTime.
- *2*: Alarm geht:
  Die Alarmvariable erhält den Wert *0* und den Zeitstempel aus TimeAck.AmlDate und TimeAck.AmlTime.

HANDSHAKE

Für die Signalisierung neuer Ereignisse muss im zenon Projekt für die Handshake-Variable jedes Ereignisblocks eine zenon Variable mit dem Objekttyp **Alarm Event** angelegt werden. Über diesen Objekttyp werden in zenon Handshake-Variablen von anderen Variablen unterschieden.

Damit der Mechanismus funktioniert, müssen die Handshake-Variablen während der gesamten Laufzeit der zenon Runtime angefordert sein. Das kann zum Beispiel über die Eigenschaft **Variable ständig lesen** erfolgen.

Die Handshake-Variablen können folgende Werte annehmen:

- *0*: Gesperrt.
  Die Steuerung darf den Block nicht für neue Ereignisse verwenden.
- *1*: Lesen angefordert.
  Die Steuerung hat den Block mit neuen Werten belegt und fordert zum Lesen des Blocks auf.
- *2*: Lesen erfolgreich.
  Der Treiber hat den Block erfolgreich gelesen und ausgewertet. Die Steuerung kann den Block wieder für neue Ereignisse verwenden.
- *4*: Fehler beim Lesen.
  Tritt beim Auswerten des Blocks ein Fehler auf, setzt der Treiber die Handshake-Variable auf *4*.

ALARMVARIABLEN

Alarmvariablen dienen dazu, die Ereignisse aus der Steuerung in zenon als Alarme darzustellen. Sie werden in zenon über den Objekttyp *Alarm Variable* von anderen Variablen des Treibers unterschieden. Sie sind über die ID mit einem Ereignis in der Steuerung verknüpft. Die ID gibt den Ort und die Art des Fehlers an.

Mit jeder ID können bis zu vier Variablen verknüpft werden: Eine immer angeforderte Variable mit einem als Alarm definierten Grenzwert sowie drei weitere optionale Variablen für weitere Informationen zum Alarm. Diese Variablen können dann z.B. in **Dynamischen Grenzwerttexten** der Alarmvariablen verwendet werden.

Für die Verknüpfung müssen die Symboladressen der Variablen einem festen Format folgen:

Adresse	Typ	Verpflichtend	Beschreibung
Exception.<ID>.Alarm	*BOOL* oder numerisch	X	Signalisiert über einen als Alarm definierten Grenzwert *1* den Alarm. Ist diese Variable nicht vorhanden oder nicht angefordert, gilt das als Fehler und der Treiber setzt die Handshake-Variable auf den Wert *4*.
Exception.<ID>.Value	numerisch	--	Erhält den Wert des Feldes Value des Ereignisblocks und kann z.B. zur Angabe eines Fehlercodes verwendet werden

Adresse	Typ	Verpflichtend	Beschreibung
Exception.<ID>.Message	*STRING*	--	Erhält den Wert des Feldes Message des Ereignisblocks, und kann einen Fehlertext aufnehmen
Exception.<ID>.Info	*STRING*	--	Erhält eine Zusammenfassung des Ereignisses im Format **<ID>;<Value>;<Message>;<AlarmState>;<ComState>;<TimeEvent>;<TimeAck>** Beispiel: **300010011;300010011;exception message1;2;1;2015-07-15T06:08:22,329+2.00;2015-07-15T06:08:38,829+2.00**

Achtung: Beachten Sie die Groß-/Klein-Schreibung.

Beispiele für Symboladressen von Alarmvariablen (ID =**300010011**):

- **Exception.300010011.Alarm**
- **Exception.300010011.Value**
- **Exception.300010011.Message**
- **Exception.300010011.Info**

Die Namen der Alarmvariablen können frei vergeben werden.

VERHALTEN BEIM HERSTELLEN DER VERBINDUNG ZUR SPS

Stellt der Treiber eine Verbindung zur SPS her, versucht er den aktuellen Status der Alarmereignisse zu eruieren. Dazu werden nicht nur Alarmereignisse ausgewertet, deren Handshake-Variablen den Wert *1* (*Lesen angefordert*) haben, sondern auch jene mit dem Wert *2* (*Lesen erfolgreich*). So können die Alarmvariablen wieder auf den Zustand vor dem Beenden der Verbindung gesetzt werden. Dieses Verhalten gilt für den Start der zenon Runtime ebenso wie für das Wiederherstellen nach einem Verbindungsabbruch.

Alarmvariablen, für die keine Einträge im Ereignis-Array existieren, werden auf einen Initialwert (**0** oder leerer String) gesetzt. Dadurch erhalten eventuell anstehende Alarme den Status *gegangen*.

Achtung: Für eine korrekte Auswertung ist es erforderlich, dass das *Kommt*-Ereignis und das *Geht*-Ereignisse eines Alarms immer im gleichen Ereignisseintrag im Array stehen.

8.5 RDA

PROBLEMSTELLUNG

Eigentlich erfordert der RDA-Mechanismus von zenon einen zusammenhängenden Block in der Steuerung, der über einen Offset adressierbar ist. Daher ist RDA in der Regel bei Treibern mit symbolischer Adressierung nicht möglich.

RDA Header muss in der Steuerung immer Byte-Order *Intel* projektiert sein.

LÖSUNG

Der **3S_V3** Treiber nutzt eine Eigenschaft des PLC-Handlers, die es erlaubt den gesamten binären Datenblock von Variablen mit komplexen Typen (Strukturen, Arrays) als Ganzes über den Basisnamen zu lesen.

CODESYS PROJEKTIERUNG

Damit die Basisnamen für komplexe Typen vom 3S_V3 Treiber korrekt gelesen werden können, muss in der Codesys Symbolkonfiguration der Eintrag **Sammeleinträge ausgeben** (**Export data entries**) aktiviert sein.

Führen Sie dazu in Codesys folgende Schritte durch:

1. Öffnen Sie Ihr Projekt im Programm Codesys.
2. Wählen Sie im Knoten **Globale Variablen** den Eintrag **Arbeitsbereich**
   Der Dialog **Optionen** wird geöffnet.
3. Wählen Sie im Dialog die Kategorie **Symbolkonfiguration**.
4. Klicken Sie die Schaltfläche **Symbolfile konfigurieren...**
   Der Dialog **Objektattribute setzen** wird geöffnet.
5. Aktivieren Sie in diesem Dialog die Checkbox **Sammeleinträge ausgeben** für die gewünschte Struktur, welche für RDA verwendet wird.
6. Bestätigen Sie die Projektierung mit Klick auf die Schaltfläche **OK**.

> **Info**
>
> Achten Sie beim Projektieren von RDA auf die Blockgröße. Diese sollte unter 95000 Byte liegen.
>
> Hintergrund: Beim synchronen Lesen der Daten darf die Kommunikation mit der SPS nicht unterbrochen werden. Denn bei einer Unterbrechung kann der RDA-Trigger nicht zurückgesetzt werden. zenon prüft beim Anfordern der Daten die Größe des RDA-Blocks. Überschreitet diese die in einem Zug übertragbare Größe, wird die Variable nicht angefordert.

RDA-PROGRAMMIERUNG

Der Treiber ermittelt den Basisnamen, indem er vom Symbolnamen der Triggervariable den Teil nach dem letzten Punkt abtrennt. Dieser Teil wird als Name verwendet. Damit das funktioniert, muss die Triggervariable das erste Element einer Struktur sein. Der Aufbau der restlichen Struktur ist beliebig. Beim Lesen der Gesamtstruktur müssen nur die eingelesenen binären Daten dem Aufbau der RDA-Daten laut zenon Dokumentation (archivserver.chm::/28257.htm) entsprechen.

BEISPIELE

RDA-TYP 3:

```
TYPE RDA_DATA_3 :                          (* Struktur für RDA-Typ-3-Nutzdaten *)
  STRUCT
    Value : DINT;                          (* Wert
    TimeStamp : ARRAY[0..7] OF BYTE; (* Zeitstempel (Jahr, Monat, Tag, Stunde, Minute, Sekunde, 1/100
Sekunde, Reserve) *)
  END_STRUCT
END_TYPE

TYPE RDA_3 :                               (* Struktur für RDA-Typ 3 *)
  STRUCT
    Trigger : DINT;                        (* Triggervariable *)
    Count : UDINT;                         (* Anzahl der Datensätze *)
    Cycle : UDINT;                         (* Zykluszeit in [ms] (nur Typ für 1 und 4 relevant) *)
    RDA_Type : UDINT;                      (* RDA-Typ, 1 - 4 *)
    Oldest : UDINT;                        (* Index des ältesten Wertes (Platzhalter aus
Kompatibilitätsgründen, nur für Typ 1 relevant) *)
    Data : ARRAY[0..19] OF RDA_DATA_3;     (* Nutzdaten *)
  END_STRUCT
END_TYPE
```

RDA-TYP 4:

```
TYPE RDA_4 :                               (* Struktur für RDA-Typ 4 *)
  STRUCT
    Trigger : DINT;                        (* Triggervariable *)
    Count : UDINT;                         (* Anzahl der Datensätze *)
    Cycle : UDINT;                         (* Zykluszeit in [ms] (nur Typ für 1 und 4 relevant *)
    RDA_Type : UDINT;                      (* RDA-Typ, 1 - 4 *)
    Oldest : UDINT;                        (* Index des ältesten Wertes (Platzhalter aus
Kompatibilitätsgründen, nur für Typ 1 relevant) *)
```

```
    TimeStamp : ARRAY[0..7] OF BYTE; (* Zeitstempel des ersten Wertes (Jahr, Monat, Tag, Stunde,
Minute, Sekunde, 1/100 Sekunde, Reserve) *)
    Data : ARRAY[0..19] OF DINT;            (* Nutzdaten *)
  END_STRUCT
END_TYPE
```

Beispiel

Ist zum Beispiel im Codesys Projekt eine Variable **RDA_Test** vom Typ *RDA_4* angelegt, dann wird in zenon eine Variable mit der symbolischen Adresse *RDA_Test.Trigger* angelegt, und **HD aktiv** und **Nachgeführte Werte** eingestellt. Ändert die Variable den Wert von *0* auf *1*, wird vom Symbolnamen *RDA_Test.Trigger* der Teil *.Trigger* abgeschnitten, und **RDA_Test** als binärer Datenblock eingelesen. Auf diesem Datenblock erfolgt dann die RDA-Bearbeitung.

8.6 Echtzeitfähigkeit

Der **3S_V3** Treiber ist echtzeitfähig.
Dabei ist zu beachten:

- Das Protokoll liefert nur sekundengenaue Zeitstempel.
- Die Zeit von SCADA-System und SPS muss manuell synchronisiert werden.

9 Funktion Treiberkommandos

Die zenon Funktion **Treiberkommandos** dient dazu, Treiber über zenon zu beeinflussen.
Mit einem Treiberkommando können Sie einen Treiber:

- starten
- stoppen
- in einen bestimmten Treibermodus versetzen
- zu bestimmten Aktionen veranlassen

Hinweis: Dieses Kapitel beschreibt Standardfunktionalitäten, die für die meisten zenon Treiber gültig sind.
Nicht alle hier beschriebenen Funktionalitäten stehen für jeden Treiber zur Verfügung. Zum Beispiel enthält ein Treiber, der laut Datenblatt keine Modemverbindung unterstützt, auch keine Modem-Funktionalitäten.

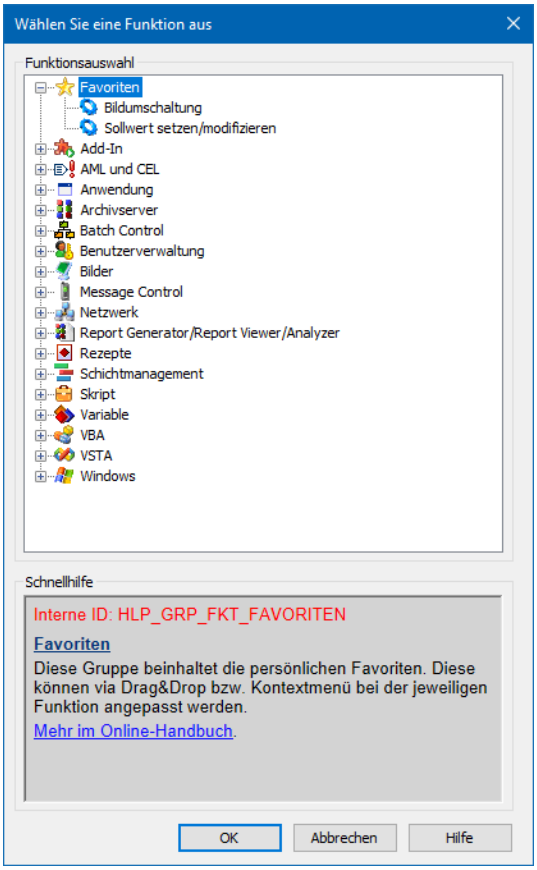
> **Achtung**
>
> Die zenon Funktion **Treiberkommandos** ist nicht ident mit den Treiberkommandos, die bei Energy-Treibern in der Runtime ausgeführt werden können!

PROJEKTIERUNG DER FUNKTION

Die Projektierung erfolgt über die Funktion **Treiberkommandos**.
Um die Funktion zu projektieren:

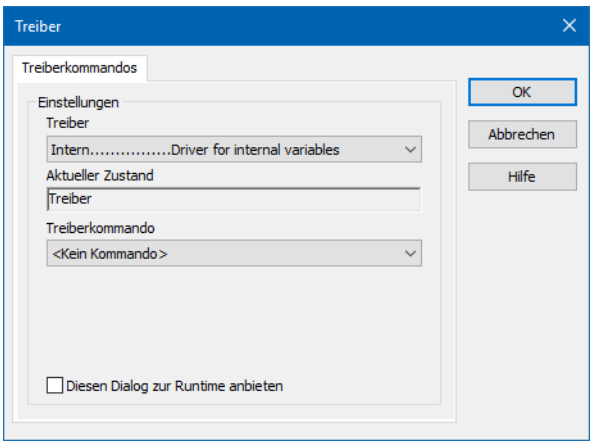
1. Legen Sie im zenon Editor eine neue Funktion an.

   Der Dialog zur Auswahl einer Funktion wird geöffnet.



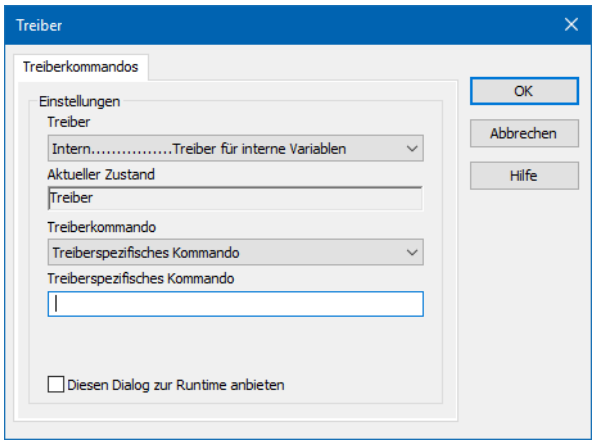
2. Navigieren Sie zum Knoten **Variable.**
3. Wählen Sie den Eintrag **Treiberkommandos**.

Der Dialog zur Konfiguration wird geöffnet.



4. Wählen Sie den gewünschten Treiber und das benötigte Kommando aus.
5. Schließen Sie den Dialog mit Klick auf **OK** und stellen Sie sicher, dass die Funktion in der Runtime ausgeführt wird.
   Beachten Sie die Hinweise im Abschnitt **Funktion Treiberkommandos im Netzwerk**.

DIALOG TREIBERKOMMANDOS



Option	Beschreibung
Treiber	Auswahl des Treibers aus der Dropdownliste. Diese enthält alle im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, der vom System gesetzt wird. In aktuellen Versionen ohne Funktion.
Treiberkommando	Auswahl des gewünschten Treiberkommandos aus Dropdownliste. Details zu den konfigurierbaren Treiberkommandos siehe Abschnitt **Verfügbare Treiberkommandos**.

Option	Beschreibung
Treiberspezifisches Kommando	Eingabe eines für den gewählten Treiber spezifischen Kommandos. **Hinweis:** Nur verfügbar, wenn für die Option **Treiberkommando** der Eintrag *Treiberspezifisches Kommando* gewählt wurde.
Diesen Dialog zur Runtime anbieten	Konfiguration, ob die Konfiguration in der Runtime geändert werden kann: ▸ *Aktiv*: Dieser Dialog wird in der Runtime vor dem Ausführen der Funktion geöffnet. Die Konfiguration kann damit in der Runtime vor der Ausführung noch geändert werden. ▸ *Inaktiv*: Die Editor-Konfiguration wird in der Runtime beim Ausführen der Funktion angewendet. Default: *inaktiv*

DIALOG BEENDEN

Option	Beschreibung
OK	**Übernimmt Einstellungen und schließt den Dialog.**
Abbrechen	Verwirft alle Änderungen und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

VERFÜGBARE TREIBERKOMMANDOS

Diese Treiberkommandos stehen - abhängig vom gewählten Treiber - zur Verfügung:

Treiberkommando	Beschreibung
<Kein Kommando>	Es wird kein Kommando gesendet. Damit kann auch ein bereits bestehendes Kommando aus einer projektierten Funktion entfernt werden.
Treiber starten (Onlinemodus)	Treiber wird neu initialisiert und gestartet. **Hinweis:** Wenn der Treiber bereits gestartet wurde, muss er gestoppt werden. Erst dann kann der Treiber wieder neu initialisiert und gestartet werden.
Treiber stoppen (Offlinemodus)	Treiber wird angehalten, es werden keine neuen Daten

Treiberkommando	Beschreibung
	angenommen. **Hinweis:** Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status *Abgeschaltet* (*OFF*; Bit *20*).
Treiber in Simulationsmodus	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiber in Hardwaremodus	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiberspezifisches Kommando	Eingabe eines treiberspezifischen Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist möglich.
Treiber Sollwertsetzen deaktivieren	Sollwert setzen auf Treiber wird verhindert.
Verbindung mit Modem aufbauen	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
Verbindung mit Modem trennen	Verbindung beenden (für Modem-Treiber).
Treiber in Simulationsmodus zählend	Treiber wird in den zählenden Simulationsmodus gesetzt. Alle Werte werden mit *0* initialisiert und in der eingestellten Updatezeit jeweils um *1* bis zum Maximalwert inkrementiert und beginnen dann wieder bei *0*.
Treiber in Simulationsmodus statisch	Es wird keine Kommunikation zur Steuerung aufgebaut. Alle Werte werden mit *0* initialisiert.
Treiber in Simulationsmodus programmiert	Die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in der zenon Logic Runtime ab.

FUNKTION TREIBERKOMMANDOS IM NETZWERK

Wenn sich der Rechner, auf dem die Funktion **Treiberkommandos** ausgeführt wird, im zenon Netzwerk befindet, werden zusätzlich weitere Aktionen ausgeführt:

- Ein spezielles Netzwerkkommando wird vom Rechner zum Server des Projekts gesendet. Dieser führt dann die gewünschte Aktion auf seinem Treiber aus.
- Zusätzlich sendet der Server das gleiche Treiberkommando zum Standby des Projekts. Der Standby führt die Aktion auch auf seinem Treiber aus.

Dadurch ist gewährleistet, dass Server und Standby synchronisiert sind. Dies funktioniert nur, wenn Server und Standby jeweils eine funktionierende und unabhängige Verbindung zur Hardware haben.

10 Fehleranalyse

Sollte es zu Kommunikationsproblemen kommen, bietet dieses Kapitel Hilfe, um den Fehler zu finden.

10.1 Analysetool

Alle zenon Module wie z. B. Editor, Runtime, Treiber, usw. schreiben Meldungen in eine gemeinsame LOG-Datei. Um sie korrekt und übersichtlich anzuzeigen, benutzen Sie das Programm Diagnosis Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter **Start/Alle Programme/zenon/Tools 8.10 -> Diagviewer.**

zenon Treiber protokollieren alle Fehler in LOG-Dateien. LOG-Dateien sind Textdateien mit einer speziellen Struktur. Der Standardordner für die LOG-Dateien ist der Ordner **LOG** unterhalb des Ordners **ProgramData**, zum Beispiel:

%ProgramData%\COPA-DATA\LOG.

Achtung: Mit den Standardeinstellungen zeichnet ein Treiber nur Fehlerinformationen auf. Mit dem Diagnosis Viewer kann bei den meisten Treibern die Diagnose-Ebene auf „Debug" und „Deep Debug" erweitert werden. Damit protokolliert der Treiber auch alle anderen wesentlichen Aufgaben und Ereignisse.

Im Diagnosis Viewer kann man auch:

- neu erstellte Einträge in Echtzeit mitverfolgen
- die Aufzeichnungseinstellungen anpassen
- den Ordner, in dem die LOG-Dateien gespeichert werden, ändern

Hinweise:

1. Der Diagnosis Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.

2. Der Diagnosis Viewer zeigt in seiner Standardeinstellung nicht alle Spalten einer LOG-Datei an. Um mehr Spalten anzuzeigen, aktivieren Sie die Eigenschaft **Add all columns with entry** im Kontextmenü der Spaltentitel.
3. Bei Verwendung von reinem **Error-Logging** befindet sich eine Problembeschreibung in der Spalte **Error text**. In anderen Diagnose-Ebenen befindet sich diese Beschreibung in der Spalte **General text**.
4. Viele Treiber zeichnen bei Kommunikationsprobleme auch Fehlernummern auf, die die SPS ihnen zuweist. Diese werden in **Error text** und/oder **Error code** und/oder **Driver error parameter(1** und **2)** angezeigt. Hinweise zur Bedeutung der Fehlercodes erhalten Sie in der Treiberdokumentation und der Protokoll/SPS-Beschreibung.
5. Stellen Sie am Ende Ihrer Tests den Diagnose-Level von **Debug** oder **Deep Debug** wieder zurück. Bei **Debug** und **Deep Debug** fallen beim Protokollieren sehr viele Daten an, die auf der Festplatte gespeichert werden und die Leistung Ihres Systems beeinflussen können. Diese werden auch nach dem Schließen des Diagnosis Viewers weiter aufgezeichnet.

> **Achtung**
>
> Unter Windows CE werden aus Ressourcegründen Fehler standardmäßig nicht protokolliert.

Weitere Informationen zum Diagnosis Viewer finden Sie im Handbuch Diagnosis Viewer (main.chm::/12464.htm).

10.2 Fehlernummern

Hinweis: Die folgende Beschreibung steht nur in englischer Sprache zur Verfügung.

Fehlertext im LOG	Nr.	Bedeutung
RESULT_FAILED	*-1*	Action erroneous.
RESULT_OK	*0*	Action successful
RESULT_PLC_NOT_CONNECTED	*1*	PLC not connected
RESULT_PLC_LOGIN_FAILED	*2*	Login to PLC has failed
RESULT_PLC_NO_CYCLIC_LIST_DEFINED	*3*	No cyclic list has been found
RESULT_PLCHANDLER_INACTIVE	*4*	PLCHandler instance was set inactive
RESULT_LOADING_SYMBOLS_FAILED	*5*	Loading of the symbols has failed
RESULT_ITF_NOT_SUPPORTED	*6*	The defined communication interface is not valid or not supported
RESULT_COMM_FATAL	*7*	Communication error occurred during action

Fehlertext im LOG	Nr.	Bedeutung
RESULT_NO_CONFIGURATION	*8*	Wrong or erroneous configuration of the PLCHandler
RESULT_INVALID_PARAMETER	*9*	At least one parameter is invalid
RESULT_ITF_FAILED	*10*	Communication interface could not be initialized correctly (e. g. Gateway Dlls not available)
RESULT_NOT_SUPPORTED	*11*	Method not yet supported resp. implemented for this configuration (e. g. not supported for this interface)
RESULT_EXCEPTION	*12*	Handled exception in a low layer occurred during action
RESULT_TIMEOUT	*13*	Timeout exceeded
RESULT_STILL_CONNECTED	*14*	PLC already connected (at a further ::Connect() call)
RESULT_RECONNECTTHREAD_STILL_ACTIVE	*15*	Reconnect Thread already active (started at a further ::Connect() call)
RESULT_PLC_NOT_CONNECTED_SYMBOLS_LOADED	*16*	No connection to the PLC, but symbols available offline
RESULT_NO_UPDATE	*17*	Asynchronous operation (e. g. cyclic read of variables) has not yet finished
RESULT_OCX_CONVERSION_FAILED	*18*	Error during conversion of values inside the PLCHandler's ActiveX interface occurred
RESULT_TARGETID_MISMATCH	*19*	PLC does not match to the passed target id etc.
RESULT_NO_OBJECT	*20*	No object found for the required action (e. g. tried to get an element beyond the end of the list)
RESULT_COMPONENTS_NOT_LOADED	*21*	PLCHandler instantiation has failed, because of missing components
RESULT_BUSY	*22*	Last action still in progress, cannot start the required one
RESULT_DISABLED	*23*	Feature is disabled by the configuration (e. g.

Fehlertext im LOG	Nr.	Bedeutung
		Logging)
RESULT_PLC_FAILED	*24*	Communication to the PLC was successful, but the PLC has returned a bad result
RESULT_INVALID_SYMBOL	*25*	Specified symbol does not exist on PLC
RESULT_BUFFER_TOO_SMALL	*26*	User provided buffer is too small to store the required information
RESULT_NO_PROJECT	*27*	No project/application is loaded by the PLC, but the called method requires one to be executed
RESULT_FILE_ERROR	*28*	Operation failed because of an file access error
RESULT_RETAIN_MISMATCH	*29*	Restored retain variables do not match to the project/application on the PLC
RESULT_NO_ACCESS_RIGHTS	*30*	Operation was denied by the PLC because of missing access rights
RESULT_DUPLICATE_PLC_NAME	*31*	Connect by PLC name failed, because there exist several PLCs with the specified name in the PLC network
RESULT_SIZE_MISMATCH	*32*	Service does not succeed because of a size mismatch

Info

Zur Anzeige dieser LOG-Nachrichten wählen Sie im Diagnosis Viewer für die Client-Konfiguration des Treibers folgende Konfiguration:

- Module **[Driver]**, Messagelevel **[Internal]**

10.3 Checkliste

Hinweise zur Verbindungsherstellung und Fehlereingrenzung:

VERBINDUNG ZWISCHEN ZENON UND EINER ABB SPS ÜBER CODESYS GATEWAY

Beachten Sie bei der Herstellung einer Verbindung zwischen zenon und einer ABB SPS mit Hilfe des CoDeSys Gateways:

1. In der Registry muss die Spezifikation des L2 ABB Protokolls enthalten sein.

   Diese Spezifikation wird normalerweise bei der Installation des ABB Automation Builders eingetragen. Sollte diese Installation nicht durchgeführt worden sein, können folgende Einträge in die Registry manuell erfolgen:

   - **Key**: *[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\3S-Smart Software Solutions GmbH\Gateway Server\Drivers\ABB Tcp/Ip Level 2 AC]*
   - **Path**: *GDrvABBTcpIpL2X.dll*
   - **Address**: *192.168.0.10*
   - **Motorola byteorder**: *Yes*
   - **Port**: *dword:000004b0*
2. Die DLL **GDrvABBTcpIpL2X.dll** muss sich im selben Ordner wie die **Gateway.exe** befinden. Die DLL muss vom Hersteller ABB direkt bezogen werden.

ALLGEMEINE FEHLERSUCHE

- Ist die Steuerung an die Stromversorgung angeschlossen?
- Analyse mit Hilfe des Diagnosis Viewers (auf Seite 75):
  -> Welche Meldungen werden angezeigt?
- Sind die Teilnehmer im **TCP/IP**-Netz verfügbar?
- Kann die Steuerung über den Befehl *Ping* erreicht werden?

  Ping: **Kommandozeile öffnen -> ping <IP-Adresse> (z. B.: ping 192.168.0.100) -> Eingabe-Taste drücken.**

  Kommt eine Antwort mit Zeitangabe oder ein Timeout?
- Kann die Steuerung auf dem entsprechenden Port über *Telnet* erreicht werden?

  Telnet: **Kommandozeile öffnen, telnet <IP-Adresse Port-Nummer> eingeben (z. B. für Modbus: telnet 192.168.0.100 502) -> Eingabe-Taste drücken .**

  Wird der Bildschirm schwarz, konnte eine Verbindung aufgebaut werden.
- Wurde die Netzadresse in den Adresseigenschaften der Variable korrekt eingestellt?
  - Stimmt die Adressierung mit der Konfiguration im Treiberdialog überein?
  - Entspricht die Netzadresse der Adresse der Zielstation?

- Wird in der Variable der richtige Objekttyp verwendet?
  Beispiel: Variablen vom Treiberobjekttyp **Kommunikationsdetails** sind reine Statistikvariablen und kommunizieren nicht mit der Steuerung.
  Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 53).
- Stimmt die Offset-Adressierung der Variable mit der in der Steuerung überein?

MANCHE VARIABLEN MELDEN INVALID

- INVALID Bits beziehen sich immer auf eine Netzadresse.
- Mindestens eine Variable der Netzadresse ist gestört.

WERTE WERDEN NICHT ANGEZEIGT, ZAHLENWERTE BLEIBEN LEER

Treiber läuft nicht. Überprüfen Sie die:

- Installation von zenon
- Installation des Treibers
- Installation aller Komponenten
  -> Achten Sie auf Fehlermeldungen beim Start der Runtime.

VARIABLEN WERDEN MIT BLAUEM PUNKT ANGEZEIGT

Die Kommunikation im Netzwerk ist gestört:

- Bei einem Netzwerkprojekt:
  Läuft das Netzwerkprojekt auch auf dem Server?
- Bei einem Standalone-Projekt oder Netzwerkprojekt, das auf dem Server läuft:
  Deaktivieren Sie die Eigenschaft **Nur von Standby Server anfordern** im Knoten **Treiber Anbindung/Adressierung**.

WERTE WERDEN FALSCH ANGEZEIGT

Überprüfen Sie die Angaben zur Berechnung im Knoten **Wertberechnung** der Variablen-Eigenschaften?

TREIBER FÄLLT SPORADISCH AUS

Analyse mit Hilfe des Diagnosis Viewers (auf Seite 75):
-> Welche Meldungen werden angezeigt?