# zenon Treiber Handbuch

## BeckhNG

v.8.00

©2018 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. BeckhNG ..... 5**

**3. BECKHNG - Datenblatt ..... 6**

**4. Treiber-Historie ..... 8**

**5. Voraussetzungen ..... 8**

5.1 PC ..... 8

5.2 Steuerung ..... 9

**6. Konfiguration ..... 10**

6.1 Anlegen eines Treibers ..... 10

6.2 Einstellungen im Treiberdialog ..... 13

6.2.1 Allgemein ..... 14

6.2.2 Beckhoff-Einstellungen ..... 18

**7. Variablen anlegen ..... 21**

7.1 Variablen im Editor anlegen ..... 21

7.2 Adressierung ..... 24

7.3 Treiberobjekte und Datentypen ..... 28

7.3.1 Treiberobjekte ..... 28

7.3.2 Zuordnung der Datentypen ..... 30

7.4 Variablen anlegen durch Import ..... 32

7.4.1 XML Import ..... 32

7.4.2 DBF Import/Export ..... 33

7.4.3 Online Import ..... 39

7.5 Kommunikationsdetails (Treibervariablen) ..... 41

**8. Treiberspezifische Funktionen ..... 47**

**9. Funktion Treiberkommandos ..... 50**

**10. Fehleranalyse ..... 53**

10.1 Analysetool ..... 53
10.2 Checkliste ..... 55

# 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. BeckhNG

### GETESTET MIT FOLGENDER HARD- UND SOFTWARE

TwinCAT SoftSPS 2.9 Build 1031

**TESTAUFBAU**

zenon Projekt mit einem Treiber und Kommunikation zu 4 TwinCAT Soft-SPS und einem Beckhoff BC Buscontroller. Eine TwinCAT SoftSPS läuft lokal auf dem PC wo das zenon Projekt läuft. Eine andere auf einem zweiten PC: Die zwei anderen TwinCAT SoftSPSn laufen auf zwei Beckhoff CX1000 CE Terminals.

# 3. BECKHNG - Datenblatt

| **Allgemein:** | |
|---|---|
| Treiberdateiname | BECKHNG.exe |
| Treiberbezeichnung | Beckhoff TwinCat NG Treiber |
| Steuerungs-Typen | TwinCAT Soft SPS 2.6 und höher - auch für Twincat v3 |
| Steuerungs-Hersteller | Beckhoff; |

| **Treiber unterstützt:** | |
|---|---|
| Protokoll | TC-ADS; |
| Adressierung: Adress-basiert | X |
| Adressierung: Namens-basiert | X |
| Kommunikation spontan | -- |
| Kommunikation pollend | X |
| Online Browsing | X |
| Offline Browsing | X |
| Echtzeitfähig | -- |
| Blockwrite | -- |
| Modemfähig | -- |
| Serielles Logging | -- |
| RDA numerisch | X |
| RDA String | -- |

| | |
|---|---|
| Hysterese | -- |
| erweiterte API | -- |
| Unterstützung von Statusbit **WR-SUC** | X |
| alternative IP-Adresse | -- |

| **Voraussetzungen:** | |
|---|---|
| Hardware PC | -- |
| Software PC | Die TC-ADS Software muss installiert sein. (TC-ADS befindet sich auf der Installations CD). TwinCAT muss nicht auf dem selben Rechner installiert sein. Der Einsatz von TwinCAT CP wird empfohlen. |
| Hardware Steuerung | TwinCAT Soft SPS auf PC und CE |
| Software Steuerung | -- |
| Benötigt v-dll | X |

| **Plattformen:** | |
|---|---|
| Betriebssysteme | Windows CE 6.0, Embedded Compact 7; Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server 2012R2, Server 2016; |
| CE Plattformen | x86; ARM; |

# 4. Treiber-Historie

| Datum | Treiberversion | Änderung |
|---|---|---|
| 07.07.08 | 2100 | Treiberdokumentation wurde neu erstellt |
| 28.11.13 | 7.11.0.9047 | Blockarrays (auf Seite 47) werden gelesen. |

**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

Um den BeckhNG Treiber mit zenon verwenden zu können müssen folgende Voraussetzungen erfüllt sein:

- Die Beckhoff TwinCat ADS Communication Library muss am Rechner installiert sein. Installieren Sie dazu die Datei **TcAdsDll.dll** in den zenon Programmordner. Sie finden diese Datei auf dem zenon Installationsmedium.
- Die Treiber **BeckhNG.EXE** Datei und die **BeckhNGV.dll**-Datei müssen ebenfalls im zenon Programmordner vorhanden sein.

Achten Sie darauf, die benötigten Dateien in dem entsprechenden Programmordner abzulegen:

- 32-Bit Betriebssysteme:
  ...\Program Files (x86)\COPA-DATA\zenon 800 SP0
- 64-Bit Betriebssysteme:
  ...\Programme\COPA-DATA\zenon 800 SP0

Die einzelnen SPS-Stationen müssen dem Betriebssystem bekannt gegeben werden

- Definieren Sie dazu die Stationen mit dem **TwinCAT AMS Router**.
- Notwendige Einträge pro Station:
  - I-Adresse
  - AS NET-ID
  - Portnummer
  - Kommunikationskanal

Wird auf dem Rechner der **TwinCAT AMS Router** nicht verwendet, kann für die notwendige Konfiguration eine minimale Installation von **TWIN CAT (TwinCAT CP)** verwendet werden.

Alternativ können die Stationen auch mit dem Tool **TcAmsRemoteMgr.exe** dem Betriebssystem bekanntgegeben werden. Dabei ist jedoch zu beachten, dass für die Kommunikation nicht das im zenon Treiberdialog konfigurierbare AMS-Timeout, sondern das im Netz gültige NETZ-Timeout verwendet wird. Bei einer Verbindungsstörung von mehreren Stationen kann es hier zu einer langen Fehlerwartezeit kommen.

**Info**

*Der Treiber ist unter WinCE verfügbar.*

*Der Treiber kann unter WinCE nur einmal je Projekt verwendet werden.*

## 5.2 Steuerung

In TwinCAT PLC Control in den Projekteigenschaften müssen unter "Symbolkonfiguration" beide Optionen zur Erzeugung der Symboldateien aktiviert werden. Mit diesem Eintrag wird die Symboldatei (*.tpy) erzeugt welche in zenon zur Import der Variablen verwendet wird.

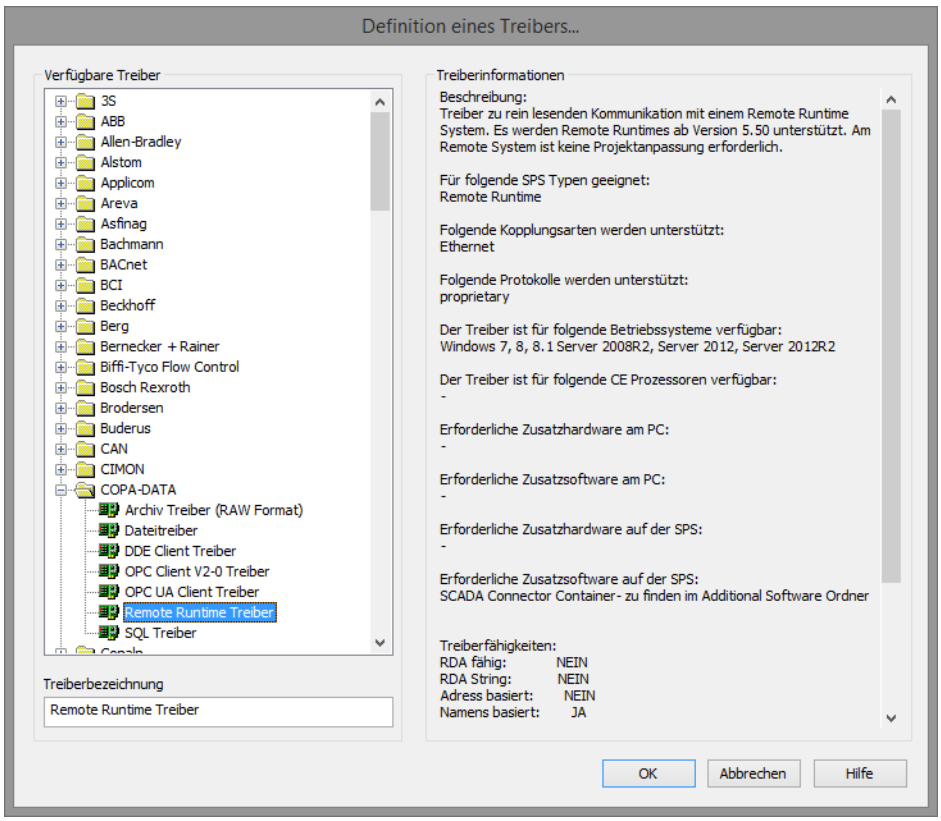
# 6. Konfiguration

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.<br>Die Darstellung erfolgt in einer Baumstruktur:<br>`[+]` erweitert die Ordnerstruktur und zeigt die darin enthaltenen Treiber.<br>[-] reduziert die Ordnerstruktur<br>Default: `keine Auswahl` |
| **Treiberbezeichnung** | Eindeutige **Bezeichnung** des Treibers.<br>Default: `leer`<br>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.<br>Default: `leer`<br>Nach Auswahl eines Treibers werden in diesem 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].`

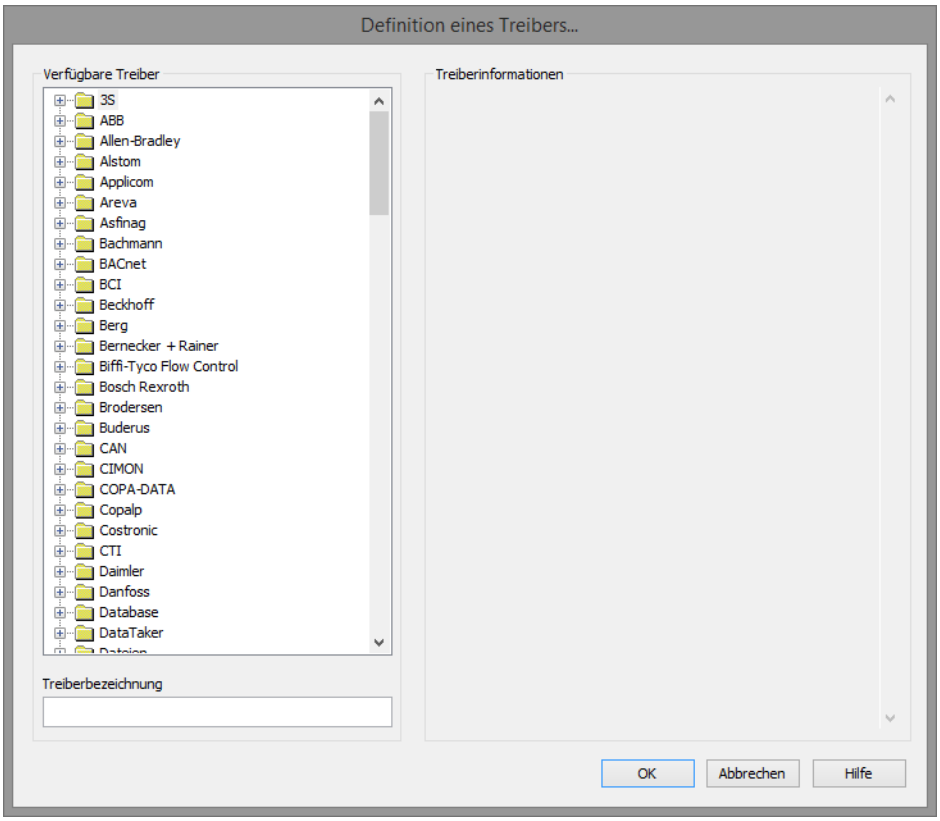
### TREIBER NEU ANLEGEN

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:

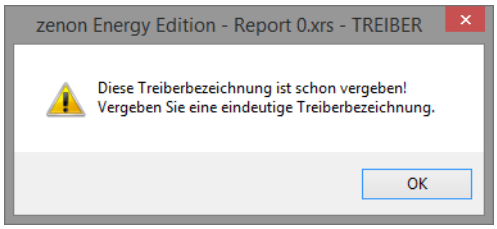
- Die **Treiberbezeichnung** muss eindeutig sein.
  Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
  Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.
- 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**

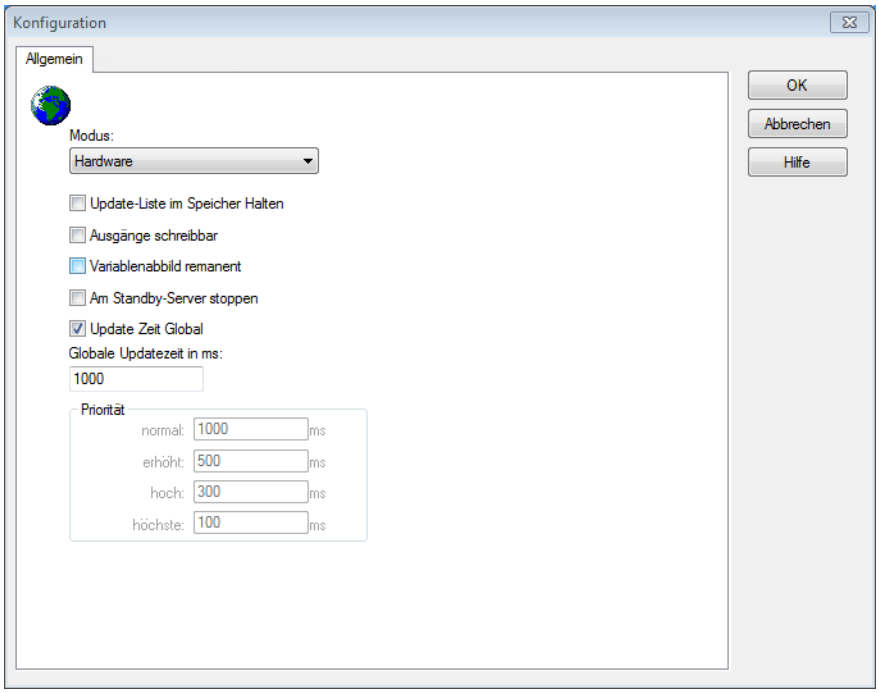
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

## 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<br>▸ `Hardware`:<br>Die Verbindung zur Steuerung wird hergestellt.<br>▸ Simulation - statisch:<br>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.<br>▸ `Simulation - zählend`:<br>Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus zählt der Treiber die Werte innerhalb ihres Wertebereichs automatisch hoch.<br>▸ `Simulation - programmiert`:<br>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.<br>Details siehe Kapitel Treibersimulation.<br>(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.<br>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`:<br>Ausgänge können beschrieben werden.<br>▸ `Inaktiv`:<br>Das Beschreiben der Ausgänge wird unterbunden.<br>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.<br>Grundvoraussetzung: Die Variable muss einen gültigen Wert und |

| | |
|---|---|
| | Zeitstempel besitzen.<br>Das Variablenabbild wird im Modus Hardware gespeichert wenn:<br>▸ einer der Status S_MERKER_1(0) bis S_MERKER8(7), REVISION(9), AUS(20) oder ERSATZWERT(27) aktiv ist<br>Das Variablenabbild wird immer gespeichert wenn:<br>▸ die Variable vom Objekttyp **Treibervariable** ist<br>▸ der Treiber im Simulationsmodus läuft. (nicht programmierte Simulation)<br>Folgende Status werden beim Start der Runtime nicht restauriert:<br>▸ `SELECT(8)`<br>▸ `WR-ACK(40)`<br>▸ `WR-SUC(41)`<br>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.<br>**Achtung:** Ist diese Option aktiv, ist die lückenlose Archivierung nicht mehr gewährleistet.<br>▸ `Aktiv`:<br>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.<br>Default: `inaktiv`<br>**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:<br>▸ `Aktiv`:<br>Die eingestellte **Globale Update Zeit** wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität wird nicht verwendet. |

| | |
|---|---|
| | ▸ Inaktiv:<br>Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet.<br><br>**Ausnahmen:** Spontane Treiber ignorieren diese Option. Sie nutzen in der Regel die kürzest mögliche Update Zeit.<br>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.<br><br>Die Zuordnung der Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften.<br>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.<br><br>**Achtung:** Prioritätsklassen werden nicht von jedem Treiber unterstützt, z.B. von spontan kommunizierenden zenon Treibern. |

### DIALOG BEENDEN

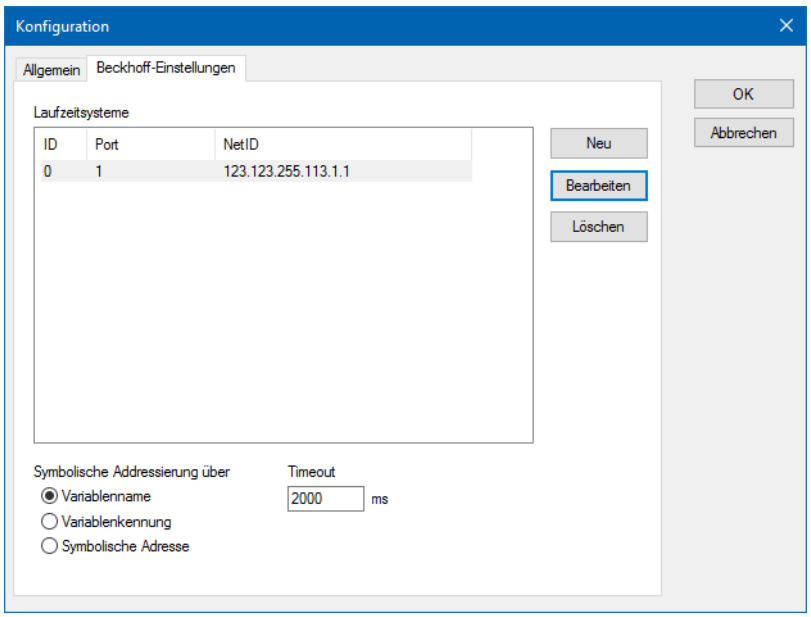
| Option | Beschreibung |
|---|---|
| **OK** | Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog. |
| **Abbrechen** | Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog. |
| **Hilfe** | Öffnet die Online-Hilfe. |

### 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 Beckhoff-Einstellungen



| Parameter | Beschreibung |
|---|---|
| **Laufzeitsysteme** | Diese Liste zeigt alle definierten Laufzeitsysteme mit **ID**, **Port** und **NetID** an.<br><br>Die definierten Laufzeitsysteme werden von den Variablen über die ID angesprochen. Dazu wird die **ID** als **Netzadresse** der entsprechenden Variable eingetragen. Diese Laufzeitsysteme müssen über die TwinCAT Software **AMS-ROUTER** oder **TcAmsRemoteMgr.exe** dem Betriebssystem bekannt gegeben werden.<br><br>Mit den Schaltflächen **Neu**, **Bearbeiten** und **Löschen** können die Einstellungen verändert werden. |
| **Symbolische Adressierung über** | Definiert, woher die Variablen-Adressierung über Symbolnamen die Symbolinformationen bezieht.<br>Auswahl über Radiobuttons:<br><br>▸ **Variablenname**:<br>Symbolinformationen kommen aus der Eigenschaft **Name** der Variablen<br><br>▸ **Variablenkennung**:<br>Symbolinformationen kommen aus der Eigenschaft **Kennung** der zenon Variablen. Dazu muss der Symbolname mit dem gleichen Präfix versehen werden wie der Variablenname. Z. B.: `S0_.Symbolname`<br>wobei `S0_` für die **Stationsnummer** `0` steht.<br><br>▸ **Symbolische Adresse**:<br>Symbolinformationen kommen aus der Eigenschaft **Symbolische Adresse** der Variablen.<br><br>Default: **Variablenname**<br><br>Hinweis: Diese Einstellung wird auch beim Online-Import berücksichtigt.<br><br>Details siehe Kapitel **Adressierung** (auf Seite 24). |
| **Timeout** | Eingabe der AMS-Timeout; Zeit in Millisekunden.<br>Für AMS-Router kann der Timeout auf `2000 ms` eingestellt werden.<br><br>Default: `2000`<br><br>Tritt ein Kommunikationsfehler mit der TwinCAT SPS auf (z. B. die Steuerung antwortet nicht), so wird die Kommunikation nach dieser Zeit unterbrochen und der Treiberstatus geht auf den Zustand `gestört`.<br><br>Siehe auch Kapitel **Voraussetzungen** (auf Seite 8). |
| **Neu** | Öffnet den Dialog zum Erstellen eines neuen Laufzeitsystems. |

| | |
|---|---|
| **Bearbeiten** | Öffnet den Dialog zum Bearbeiten des ausgewählten Laufzeitsystems. |
| **Löschen** | Löscht das ausgewählte Laufzeitsystem aus der Liste. |

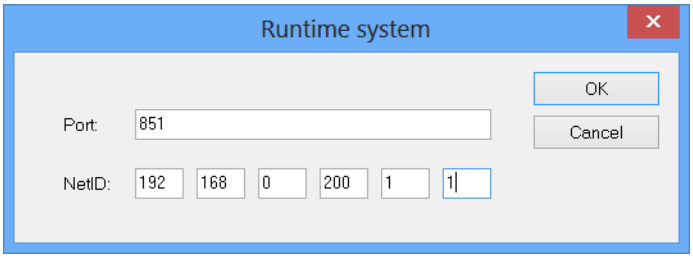
### DIALOG BEENDEN

| Option | Beschreibung |
|---|---|
| **OK** | Übernimmt Einstellungen und schließt den Dialog. |
| **Abbrechen** | Verwirft alle Änderungen und schließt den Dialog. |

## LAUFZEITSYSTEM ERSTELLEN/BEARBEITEN

Um ein Laufzeitsystem zu erstellen:

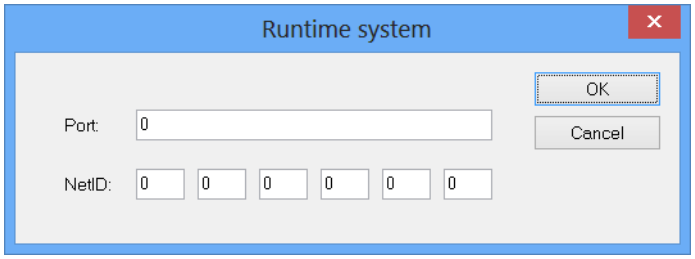
1. Klicken Sie in der Registerkarte **Beckhoff-Einstellungen** auf die Schaltfläche **Neu**.
   Der Dialog für die Konfiguration wird geöffnet.
2. Tragen Sie **Port** und **NetID** ein.




Um einen Eintrag zu bearbeiten:

1. Wählen Sie das gewünschte Laufzeitsystem in der Liste aus.
2. Klicken Sie auf auf die Schaltfläche **Bearbeiten**.
   Der Dialog für die Konfiguration wird geöffnet.
3. Bearbeiten Sie **Port** und **NetID**.

### DIALOG LAUFZEITSYSTEM



| Parameter | Beschreibung |
|---|---|
| **Port** | Eingabe des Ports. |
| **NetID** | Eingabe der Netzadresse mit ID. |
| **OK** | Übernimmt Einstellungen und schließt den Dialog. |
| **Cancel** | Verwirft Einstellungen und schließt Dialog. |

**Info**

*Standard ADS-Ports:*

- TwinCAT 2: 801
- TwinCAT 3: 851
- Buscontrollern (z. B. BC9000): 800

# 7. Variablen anlegen

So werden Variablen im zenon Editor angelegt:

## 7.1 Variablen im Editor anlegen

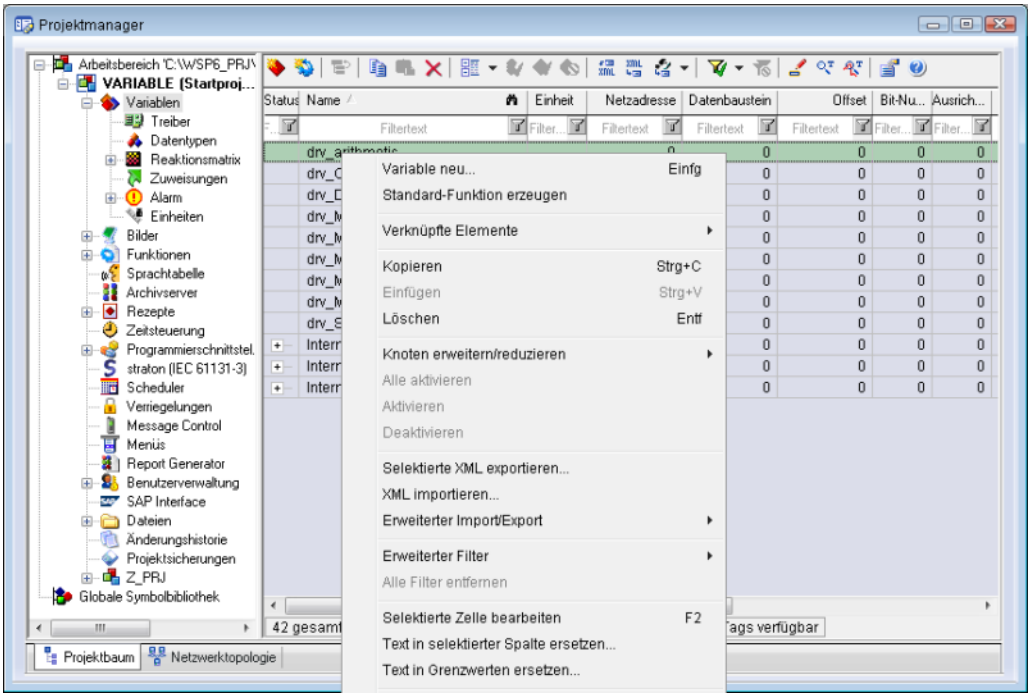
Variablen können angelegt werden:

- 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:

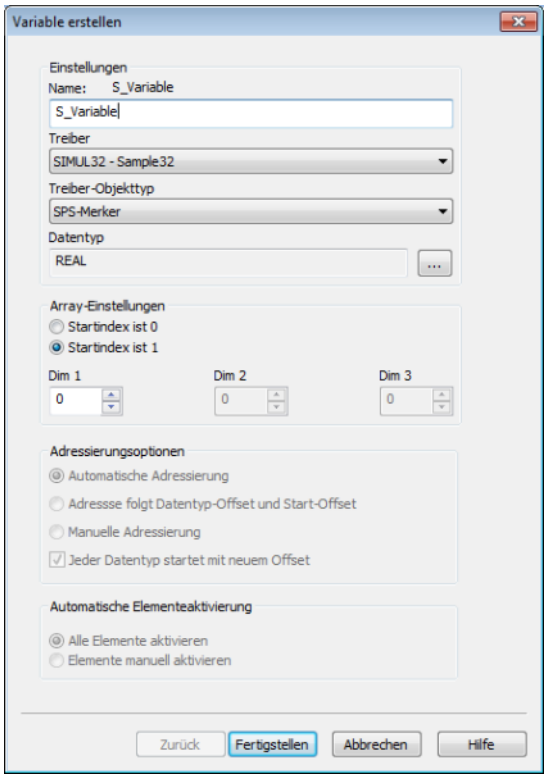
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.




Der Dialog zur Konfiguration der Variable wird geöffnet.

2. Konfigurieren Sie die Variable.

3. Welche Einstellungen möglich sind, hängt ab vom Typ der Variablen.




| 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.<br><br>Maximale Länge: 128 Zeichen<br><br>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.<br>Hinweis: Manche Treiber erlauben die Adressierung auch über die Eigenschaft **Symbolische Adresse**. |
| **Treiber** | Wählen Sie aus der Dropdownliste den gewünschten Treiber.<br><br>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. |

| | |
|---|---|
| **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.

### 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

### ZUGRIFFSVERFAHREN

Polling - Der Treiber fragt die Werte aus der Steuerung fortlaufend ab.

### ANZAHL DER STEUERUNGEN

Es können mit einem Treiber mehrere Steuerungen angesprochen werden.

## DIE EINSTELLUNGEN DER VARIABLE WERDEN WIE FOLGT BELEGT

| Einstellung an der Variable | Feld aus der XML Datei |
|---|---|
| **Name** | Symbolname mit Stationsprefix, z. B. „**S2_SymName**" (siehe dazu auch Adressierung über Name).<br>Die Adressierung erfolgt symbolisch über Name und wird beim online-Import entsprechend der Konfiguration automatisch gesetzt.<br>Die zweite „Kennung“ bei den Properties meint wahrscheinlich die „Betriebsmittelkennung |
| **Kennung** | Symbolname mit Stationsprefix, z. B. „**S2_SymName**" (siehe dazu auch Adressierung über Kennung).<br>Die Adressierung erfolgt symbolisch über Kennung und wird beim online-Import entsprechend der Konfiguration automatisch gesetzt.<br>Die zweite „Kennung“ bei den Properties meint wahrscheinlich die „Betriebsmittelkennung |
| **Netzadresse** | ID der entsprechenden Station aus der Treiberkonfiguration.<br>Ist keine Station mit entsprechender AmsNetID vorhanden so wird automatisch eine Neue Station angelegt.<br>(siehe dazu auch Beckhoff-Einstellungen) |
| **Treiber Anbindung/Datentyp** | Der Datentyp wird IEC konform vergeben. |
| **Treiber Anbindung/Treiberobjekttyp** | Als Treiber Objekttyp wird ein SPS-Merker verwendet. |
| **Offset** | Hier wird der Offset aus der Steuerung übernommen. |
| **iGroup** | Die Index Gruppe aus der Steuerung.<br>Wird beim online import automatisch gesetzt. |
| **BitSize** | Die BitSize aus der Steuerung.<br>Wird beim online import automatisch gesetzt.<br>Folgende BitSize muss für jeweilige Datentypen gesetzt sein:<br>8 BOOL<br>8 BYTE<br>32 DATE<br>32 DINT |

| | | |
|---|---|---|
| | 32 | DT |
| | 32 | DATE_AND_TIME |
| | 32 | DWORD |
| | 16 | INT |
| | 64 | LREAL |
| | 32 | REAL |
| | 8 | SINT |
| | 648 | STRING |
| | 32 | TIME |
| | 32 | TOD |
| | 32 | TIME_OF_DAY |
| | 32 | UDINT |
| | 16 | UINT |
| | 8 | USINT |
| | 16 | WORD |
| Betriebsmittelkennung | Der Kommentar wird als Kennung angelegt. Alle Zeilenumsprünge und Tabulatoren werden dabei entfernt. | |
| **Treiber Anbindung/Priorität** | Wird für diesen Treiber nicht verwendet. Der Treiber unterstützt keine zyklisch pollende Kommunikation in Prioritätsklassen. | |
| Symbolische Adresse | Die Adressierung erfolgt symbolisch über Symbolische Adresse und wird beim online-Import entsprechend der Konfiguration automatisch gesetzt.<br>Die zweite „Kennung“ bei den Properties meint wahrscheinlich die „Betriebsmittelkennung | |

## ÜBERSCHREIBEN DER BESTEHENDEN VARIABLEN (ERWEITERTES MERGING)

Sollte eine zu importierende Variable in zenon bereits bestehen wird wie folgt vorgegangen.

- Ist die Variable von einem anderen Treiber, wird diese nicht verändert.
- Hängt die betroffene Variable am aktuellen Treiber, so werden bei Änderungen überschrieben:
  - Datentyp

- Treiber Objekttyp
- Offset
- iGroup
- Bitsize
- Kennung

### VARIABLENADRESSIERUNG

In der zenon Runtime erfolgt die Adressierung über

- Name
- Kennung
- Adresse

Im ersten Schritt wird ermittelt, ob eine Station mittels Namen oder Kennung kommunizieren kann. Wenn diese Kommunikationsart nicht erfolgreich ist, wird der Treiber die Variablen der entsprechenden Station über die Adresse auslesen bzw. schreiben.

Ablauf:

- Existiert ein Symbol mit dem Namen, kommt ein Handle zurück über den dann kommuniziert wird.
- Existiert kein Symbol mit dem Namen in der SPS (ADS **Error 0x710**), dann:
  - wird im Diagnose-Tool gelogged, dass kein Symbol existiert
  - erhält die Variable den Status `INVALID`
  - erfolgt kein Versuch, über den Offset zu kommunizieren
- Nur wenn der Versuch, ein Handle zu holen, mit dem ADS **Error 0x701** (service not supported by server) beantwortet wird, wird versucht über den Offset zu lesen.

### NAME

Bei Adressierung über den Namen, muss der Variablenname in zenon aus einem Prefix und dem entsprechenden Symbolnamen zusammengesetzt werden.

- Prefix: **Sn**_
  - **n**: Netzadresse, da ja das gleiche Programm auf unterschiedlichen Steuerungen laufen könnte)
  - Beispiel:

    zenon: **S2_SymbolName**

    Steuerung: **SymbolName**

#### KENNUNG

Bei Adressierung über die Kennung versucht der Treiber den Symbolnamen aus der Variablenkennung zu beziehen. dazu muss:

- in der Treiberkonfiguration (auf Seite 18) die Checkbox **Kennung beinhaltet Symbolname** aktiviert sein
- der Symbolname mit dem gleichen Prefix versehen werden wie bei der Adressierung über **Name**

Der Onlineimport berücksichtigt diese Einstellung und schreibt den Symbolnamen auf die Kennung der Variablen anstelle des Variabennamens. Der Online Import kann in diesem Modus etwas langsamer ausfallen, da vor dem Import eine Zuweisungsliste von Variablennamen zu Variablenkennung erstellt werden muss. Zudem werden die Kommentare nicht mehr importiert während bei der Modus Adressierung über Variablennamen der Kommentar in die Kennung geschrieben wird.

#### ADRESSE

| Parameter | Beschreibung |
|---|---|
| **Netzadresse** | Als Netzadresse wird die ID der im Konfigurationsdialog angelegten Station (Port und AMS Netaddress) zur Auswahl angeboten. |
| **BitSize** | Grösse des Datenpunktes in Bit |
| **iGroup** | Index Gruppe in der angesprochenen Station |
| **Offset** | Der Offset innerhalb einer iGroup |

## 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 | Kommentar |
|---|---|---|---|---|---|
| **SPS-Merker** | 8 | X | X | WORD, LREAL, BOOL, DWORD, DINT, DATE_AND_TIME, REAL, UDINT, INT, DATE, TOD, UINT, SINT, USINT, TIME, STRING, LINT, ULINT | |
| **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).<br>**Hinweis**: Die Adressierung und das Verhalten ist bei den meisten zenon Treibern gleich.<br>Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 41). |

## Legende:

**X**: wird unterstützt

**--**: wird nicht unterstützt

**BEISPIELE ALLER MÖGLICHEN IEC DATENTYPEN**

| Datentyp der SPS | Datentypen in zenon |
|---|---|
| **BOOL** | BOOL |
| **BYTE** | USINT |
| **DATE** | DATE |
| **DINT** | DINT |
| **DT** | DATE_AND_TIME |
| **DATE_AND_TIME** | DATE_AND_TIME |
| **DWORD** | DWORD |
| **INT** | INT |
| **INT16** | INT |
| **LINT** | LINT |
| **LREAL** | LREAL |
| **REAL** | REAL |
| **SINT** | SINT |
| **STRING** | STRING |
| **TIME** | TIME |
| **TOD** | TOD |
| **TIME_OF_DAY** | TOD |
| **UDINT** | UDINT |
| **UINT** | UINT |
| **UINT16** | UINT |
| **ULINT** | ULINT |
| **USINT** | USINT |
| **WORD** | WORD |

## 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.

## BEISPIELE ALLER MÖGLICHEN ZENON DATENTYPEN

| SPS | zenon |
|---|---|
| **IX 1.0** | Eingangsbit Offset 1 Bit 1 |
| **I 1** | Eingangsbyte Offset 1 niederwertig |
| **MB 100** | Merkerbyte Offset 100 niederwertig |
| **MX 100.0** | Merkerbit Offset 100 Bit 0 |
| **MW 100** | Merkerwort Offset 100 |
| **Q 1** | Ausgangsbyte Offset 1 niederwertig |
| **QW 60000** | Ausgangswort Offset 60000 |
| **QX 1.1** | Ausgangsbit Offset 1 Bit 1 |

## TREIBER OBJEKTTYPEN UND UNTERSTÜTZTE IEC DATENTYPEN FÜR PROZESSVARIABLEN IN ZENON

| Treiber Objekttypen | Kanaltyp | Unterstützte Datentypen (Datenart) | Lesen | Schreiben | Kommentar |
|---|---|---|---|---|---|
| Merker | 8 | BOOL<br>BYTE<br>DATE<br>DINT<br>DT<br>DATE_AND_TIME<br>DWORD<br>INT<br>INT16<br>LINT<br>LREAL<br>REAL<br>SINT<br>STRING<br>TIME<br>TOD<br>TIME_OF_DAY<br>UDINT<br>UINT | J | J | |

| | | | | | |
|---|---|---|---|---|---|
| | | UINT16<br>ULINT<br>USINT<br>WORD | | | |

**Datenart:** Die Eigenschaft **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.

#### 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.<br>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).<br>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.<br>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) |
| **ADRESSE** | N | 5 | Offset |
| **BITADR** | N | 2 | Für Bit-Variablen: Bitadresse<br>Für Byte-Variablen: 0=niederwertig, 8=höherwertig<br>Für String-Variablen: Stringlänge (max. 63 Zeichen) |
| **ARRAYSIZE** | N | 16 | Anzahl der Variablen im Array für Index-Variablen<br>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<br>0: Sollwert setzen ist nicht erlaubt<br>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<br>setzt sich zusammen aus TREIBER-OBJEKTTYP und DATENTYP |
| **SIGMIN** | Float | 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)<br>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, 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.<br>Freier String für Export und Anzeige in Listen.<br>Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden. |
| **ADJWVBA** | L | 1 | Nichtlineare Wertanpassung:<br>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:

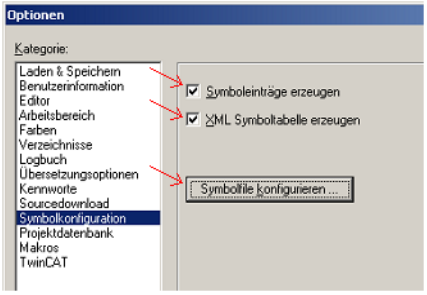
| Bezeichnung | Typ | Feldgröße | Bemerkung |
|---|---|---|---|
| **AKTIV1** | L | 1 | Grenzwert aktiv (pro Grenzwert vorhanden) |
| **GRENZWERT1** | F | 20 | technischer Wert oder ID-Nummer der verknüpften Variable für einen dynamischen Grenzwert (siehe VARIABLEx)<br>(wenn unter VARIABLEx `1` steht 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<br>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<br>(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 |
| **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.

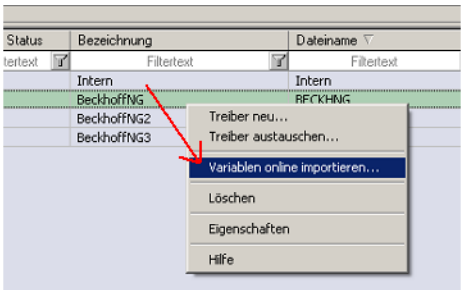
## 7.4.3 Online Import

### VARIABLEN ONLINE IMPORTIEREN

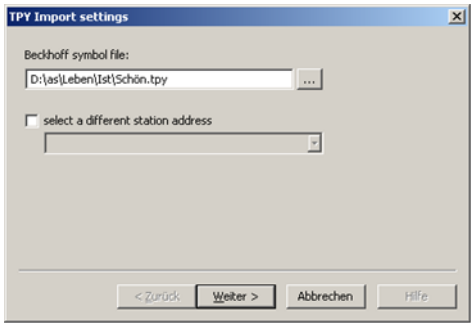
Variablen können über die Symboldatei (*.tpy) in zenon eingebunden werden. Dazu müssen im entsprechendem TwinCAT Projekt unter Optionen/Symbolkonfiguration die Checkboxen **Symboleinträge erzeugen** und **XML Symboltabelle erzeugen** aktiviert sein und das Symbolfile entsprechend konfiguriert sein (siehe Schaltfläche unterhalb).



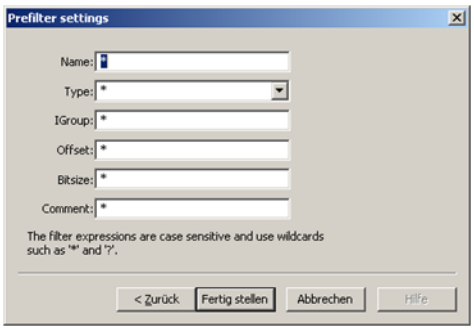

In zenon können die Variablen bequem über den Onlineimport angelegt bzw. geändert werden. Klicken sie dazu im zenon Editor mit der rechten Maustaste auf den gewünschten Treiber und wählen Sie im Kontextmenü den Punkt **Variablen online importieren...**.



Als nächstes werden Sie aufgefordert den Pfad zum Symbolfile anzugeben. Nach dem bestätigen der Auswahl mit **Fertigstellen** wird das Symbolfile geladen. Dies kann je nach Variablenanzahl bis zu mehreren Sekunden in Anspruch nehmen. In diesem Dialog kann eine bereits in die Konfiguration erstellte Station ausgewählt werden zu der die Variablen importiert werden sollen. Wird hier keine Station angegeben, wird die AMS NET-ID aus der .TPY Datei verwendet und wenn diese Station nicht vorhanden ist, eine Station angelegt.

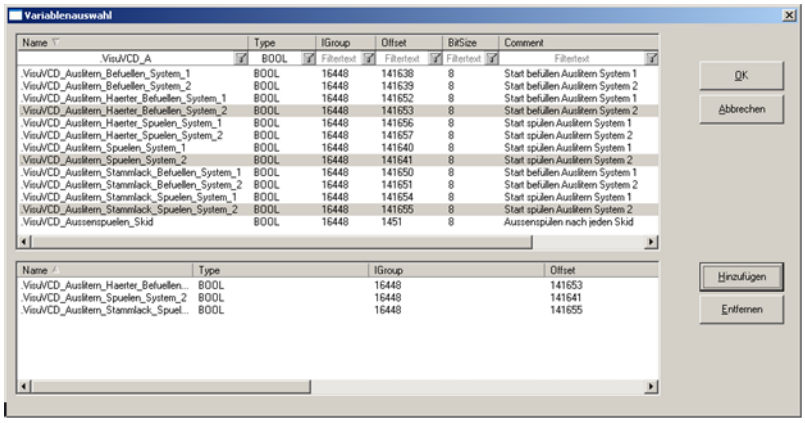



**Achtung:** Ist in der .tpy Datei keine AMS Net ID vorhanden, wird automatisch eine Station mit der AMS Net ID "0.0.0.0.0.0" erstellt. Wählen Sie beim Import manuell die richtige AMS Net ID oder passen Sie die AMS Net Id in der Treiberkonfiguration an. Eine .tpy Datei aus TwinCAT 3.x enthält keine AMS Net ID.




Anschließend ist es möglich, eine Vorfilterung durchzuführen. Hier kann z. B. auf Kommentare im TwinCAT Programm gefiltert werden, damit nur bestimmten Variablen für die Visualisierung angeboten werden. Vor allem bei großen TwinCAT Projekte mit vielen Strukturen ist es sinnvoll hier eine Vorfilterung anzuwenden.

Als Resultat wird ein Auswahldialog mit einer flachen (Strukturen und Arrays sind aufgeschlüsselt) Symbolliste angezeigt. Am Spaltenkopf kann nun beliebig gefiltert und sortiert werden. Über die Schaltflächen **Hinzufügen** bzw. **Entfernen** können Sie die Auswahl der zu importierenden Variablen verändern.




Nach dem Verlassen des Dialogs mit **Ok** werden die ausgewählten Variablen in zenon angelegt.

### Info

*Der Online-Import berücksichtigt die Einstellung **Kennung beinhaltet Symbolname** aus Treiberdialog Beckhoff-Einstellungen (auf Seite 18):*

`Aktiv`*: Der Symbolnamen wird anstelle des Variabennamens auf die Kennung der Variablen geschrieben. Der Online Import kann in diesem Modus etwas langsamer ausfallen da vor dem Import eine Zuweisungsliste von Variablennamen zu Variablenkennung erstellt werden muss. Kommentare werden nicht nicht mehr importiert.*

`Inaktiv`*: Die Adressierung erfolgt über den Variablennamen, der Kommentar wird in die Kennung geschrieben.*

## 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** (auf der Installationsmedium im Ordner `\Predefined\Variables`) verfügbar und können von dort importiert werden.

**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 werden:*

- Variablen für Modem-Informationen nur von modemfähigen Treibern unterstützt
- Treibervariablen für den Polling-Zyklus nur für rein pollenden Treibern
- verbindungsbezogene Informationen wie ErrorMSG nur von Treibern, 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. |
| 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<br>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.<br>Verbindungszustände:<br>0: Verbindung OK<br>1: Verbindung gestört<br>2: Verbindung simuliert<br>Formatierung:<br><Netzadresse>:<Verbindungszustand>;…;…;<br>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.<br>Der Zustand einer Verbindung wird nur aktualisiert, wenn eine Variable der Verbindung angemeldet ist. Ansonsten wird nicht mit der entsprechenden Steuerung 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. |
| Baudrate | UDINT | 10 | Baudrate der seriellen Schnittstelle. |
| Parity | SINT | 11 | Parität der seriellen Schnittstelle. |
| ByteSize | USINT | 14 | Bitanzahl pro Zeichen der seriellen Schnittstelle.<br><br>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 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. |
| 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:

### EINSCHRÄNKUNGEN

Bei der Kommunikation zu einem Beckhoff Buscontroller (z. B. BC9000) wird die Kommunikation zuerst über ein Handle versucht. Da der Buskontroller diese Art von Kommunikation nicht unterstützt, findet die Kommunikation verzögert statt. Anstelle des **BeckhNG** Treibers sollte hier der **BeckhBC** Treiber verwendet werden. Dieser Treiber wurde speziell für diesen Steuerungstyp entwickelt.

### BLOCKARRAYS

- Blockarrays können gelesen werden.
  - Namensbasierte Steuerungen:
    Dabei wird der Symbolname ohne Indizes verwendet.
    Beispiel: **MAIN.MyArray**. Es dürfen keine eckigen Klammern gesetzt werden.
  - Adressbasierte Steuerungen:
    Dabei wird die Adresse des ersten Index verwendet.
- Die Anzahl der Indizes muss mit denen in der Steuerung übereinstimmen.
  Beispiel:
  für **MyArray: ARRAY [0..99] OF INT;**
- In den zenon Variablen-Eigenschaften muss in der Gruppe **Zusätzliche Einstellungen** der Wert der Eigenschaft **Größe Blockarray** auf `100` eingestellt werden

### BLOCKWRITE

Der Treiber unterstützt kein Blockwrite.

### ECHTZEITSTEMPELUNG

Der Treiber unterstützt keine Echtzeitstempelung.

### FEHLERDATEI

Fehler werden in das zentrale zenon LOG geschrieben und können mit dem Diagnose Viewer ausgewertet werden.

### ERWEITERTE FEHLERDATEI

Der Treiber schreibt keine erweiterte Fehlerdatei.

## REDUNDANZ

Redundanzbetrieb ist mit diesem Treiber möglich.

## RDA

Der **BeckhNG** Treiber nützt eine Eigenschaft der ADS-Schnittsstelle, die es erlaubt, den gesamten binären Datenblock von Variablen mit komplexen Typen (Strukturen, Arrays) vollständig über den Basisnamen zu lesen.

Der Treiber ermittelt den Basisnamen, indem er vom Symbolnamen der Triggervariable den Teil nach dem letzten Punkt (**.**) abtrennt, und den so gewonnenen Namen verwendet. Damit diese Methode 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
```

Ist zum Beispiel im TwinCAT 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.

# 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

**Achtung:** Die zenon Funktion **Treiberkommandos** ist nicht ident mit Treiberkommandos, die bei Energy-Treibern zur Runtime ausgeführt werden können!

**Info**

*Dieses Kapitel beschreibt Standardfunktionalitäten, die für die meisten zenon Treiber gültig sind.*
*Aber 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.*

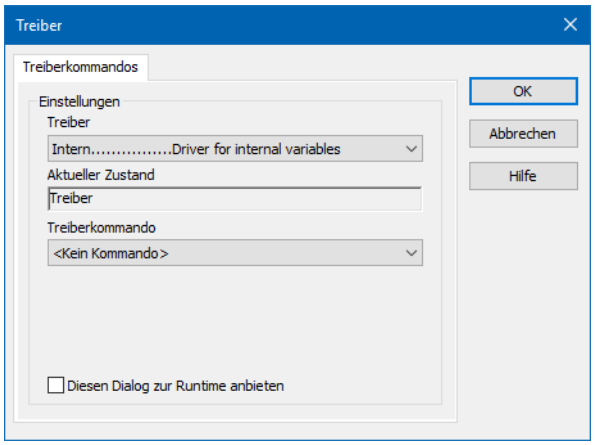
## PROJEKTIERUNG DER FUNKTION

Die Projektierung erfolgt über die Funktion **Treiberkommandos**.
Dazu:

1. Legen Sie im zenon Editor eine neue Funktion an.
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 zur Runtime ausgeführt wird.
   Beachten Sie die Hinweise im Abschnitt **Funktion Treiberkommandos im Netzwerk**.

## DIALOG TREIBERKOMMANDOS



| Option | Beschreibung |
|---|---|
| **Treiber** | Auswahl des Treibers aus der Dropdownliste.<br>Diese enthält alle im Projekt geladenen Treibern. |
| **Aktueller Zustand** | Fixer Eintrag, in aktuellen Versionen ohne Funktion. |
| Treiberkommando | Dropdownliste zur Auswahl des Kommandos: |
| <Kein kommando> | Es wird kein Kommando gesendet.<br>Damit kann auch ein bereits bestehendes Kommando aus einer projektierten Funktion entfernt werden. |
| `Treiber starten (Onlinemodus)` | Treiber wird neu initialisiert und gestartet. |
| `Treiber stoppen (Offlinemodus)` | Treiber wird angehalten, es werden keine neuen Daten angenommen.<br>**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.<br>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.<br>Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt. |
| `Treiberspezifisches Kommando` | Eingabe treiberspezifischer Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos. |
| `Treiber Sollwertsetzen aktivieren` | Sollwert setzen auf Treiber ist erlaubt. |
| `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.<br>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` | Treiber wird in den zählenden Simulationsmodus gesetzt.<br>Alle Werte werden mit `0` initialisiert. |
| `Treiber in Simulationsmodus programmiert` | Treiber wird in den zählenden Simulationsmodus gesetzt.<br>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. |
| **Diesen Dialog zur Runtime anbieten** | Dialog wird zur Runtime für Änderungen angeboten. |

#### 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, der dann die gewünschte Aktion auf seinem Treiber durchführt. 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 8.00 -> 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 Checkliste

| Problem | Diagnose | Ursache |
|---|---|---|
| Es können keine Werte von der Steuerung gelesen oder geschrieben werden. | Die Steuerung kann per 'Ping' nicht erreicht werden? | ▸ Die Steuerung ist nicht an die Stromversorgung und/oder das Netzwerk angeschlossen.<br>▸ Der PC ist nicht am Netzwerk angeschlossen.<br>▸ Die Steuerung ist angeschlossen befindet sich aber in einem anderen Subnetz und das Netzwerk Gateway ist nicht in der Steuerung eingetragen bzw. die Subnetmask ist falsch eingestellt.<br>▸ Ist die Firewall aktiviert? Als Default wird Port 801(TC2x)/851(TC3x) zur Kommunikation verwendet, zu den Ausnahmen hinzufügen. Bei individueller Portnummer entsprechend diese eintragen. |
| | Die Steuerung kann per 'Ping' erreicht werden? | ▸ Die Kommunikationsparameter AMS-NET ID, ist nicht korrekt eingestellt?<br>▸ Der Port muss entsprechend der Konfiguration der Steuerung eingestellt sein.<br>▸ Die Netzadresse in der Adressierung der Variable stimmt mit der Verbindungs ID im Treiber nicht überein.<br>▸ Die Treiberkonfigurations Datei wurde nicht auf den Zielrechner übertragen?<br>▸ Die AMS-Routen sind nicht konfiguriert. Hinweis: Am PLC System als auch am zenon Runtime System muss die Partnerstation als Route eingetragen sein. |

| | | |
|---|---|---|
| Es können bestimmte Werte von der Steuerung nicht gelesen oder geschrieben werden. | Wurde mit dem Diagnosis Viewer bereits analysiert, welche Fehler sind aufgetreten?<br>Siehe Kapitel Analysetool (auf Seite 53) | ▸ Sind die Variablen korrekt adressiert? IGroup und<br>▸ Werden die korrekten Objekttypen in der variable verwendet? Die Objekttypen legen den zu verwendeten Funktionscode im Modbus Telegramm fest. |
| Es werden falsche Werte angezeigt. | Wurde mit dem Diagnosis Viewer bereits analysiert, welche Fehler sind aufgetreten?<br>Siehe Kapitel Analysetool (auf Seite 53) | ▸ Sind die Variablen korrekt adressiert, die Symbolnamen Korrekt?<br>▸ Sind die korrekten Datentypen verwendet?<br>▸ Ist die Wertberechnung korrekt? |

- Ist der AMS-Router richtig konfiguriert?
- Sind die entsprechenden Remote Computer wechselseitig eingetragen?
- Ist der richtige Port eingestellt?