zenon
by COPA-DATA

zenon Treiber Handbuch

MODRTU32

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 MODRTU32 5

3 MODRTU32 - Datenblatt 6

4 Treiber-Historie 7

5 Voraussetzungen 8

5.1 PC 8

6 Konfiguration 8

6.1 Anlegen eines Treibers 9

6.2 Einstellungen im Treiberdialog 12

6.2.1 Allgemein 13

6.2.2 Com 17

6.2.3 Einstellungen 19

6.2.4 Verbindung TCP/IP 24

7 Variablen anlegen 27

7.1 Variablen im Editor anlegen 27

7.2 Adressierung 31

7.3 Treiberobjekte und Datentypen 32

7.3.1 Treiberobjekte 32

7.3.2 Zuordnung der Datentypen 36

7.4 Variablen anlegen durch Import 37

7.4.1 XML Import 38

7.4.2 DBF Import/Export 39

7.5 Kommunikationsdetails (Treibervariablen) 44

8 Treiberspezifische Funktionen 51

9 Funktion Treiberkommandos 51

10 Fehleranalyse 56

10.1 Analysetool 56

10.2 Fehlernummern 57

10.3 Checkliste 59

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 MODRTU32

Der Treiber **MODRTU32** ist ein Modbus Master. (Für Modbus Slave steht das Process Gateway zur Verfügung).

Folgende Protokolle werden vom Treiber unterstützt:

- *Modbus ASCII*
  für serielle Verbindungen
- *Modbus RTU*
  für serielle und für TCP/IP Verbindungen

- *Open Modbus TCP*
  für TCP/IP Verbindungen.

3 MODRTU32 - Datenblatt

Allgemein:	
Treiberdateiname	MODRTU32.exe
Treiberbezeichnung	Modbus RTU und Open Modbus TCP
Steuerungs-Typen	Alle Steuerungen, die Modbus RTU bzw. Open Modbus TCP oder Modbus ASCII (seriell) unterstützen
Steuerungs-Hersteller	ABB; Gantner; GE Automation&Controls; Modbus RTU; Mondial; Schiele; Telemecanique; Schneider; Wago; SE Elektronic; Areva; GE Multilin; CIMON

Treiber unterstützt:	
Protokoll	Modbus RTU - serial; Open Modbus TCP; Modbus RTU over TCP; Modbus ASCII - serial
Adressierung: Adress-basiert	Address based
Adressierung: Namens-basiert	--
Kommunikation spontan	--
Kommunikation pollend	X
Online Browsing	--
Offline Browsing	--
Echtzeitfähig	--
Blockwrite	X
Modemfähig	--
RDA numerisch	X
RDA String	--

Treiber unterstützt:	
Hysterese	X
erweiterte API	X
Unterstützung von Statusbit **WR-SUC**	X
alternative IP-Adresse	--

Voraussetzungen:	
Hardware PC	RS 232 Schnittstelle oder Standard Netzwerkkarte
Software PC	--
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	Treiberversion	Änderung
07.07.08	4400	Treiberdokumentation wurde neu erstellt

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 eine serielle Verbindung zur Steuerung herstellen zu können wird am Rechner ein serieller COM-Port benötigt. Wichtig dabei ist, dass der Port zur Laufzeit nicht von anderer Software (z. B. Modemtreiber, Steuerungssoftware, ...) verwendet und somit blockiert wird.

Dieser Treiber unterstützt eine Verbindung über die Standard Netzwerkkarte des PCs. Achten Sie darauf, dass die Steuerung und der PC im selben Netzwerkbereich liegen und die Subnet-Masken auf beiden Geräten dementsprechend konfiguriert sind.

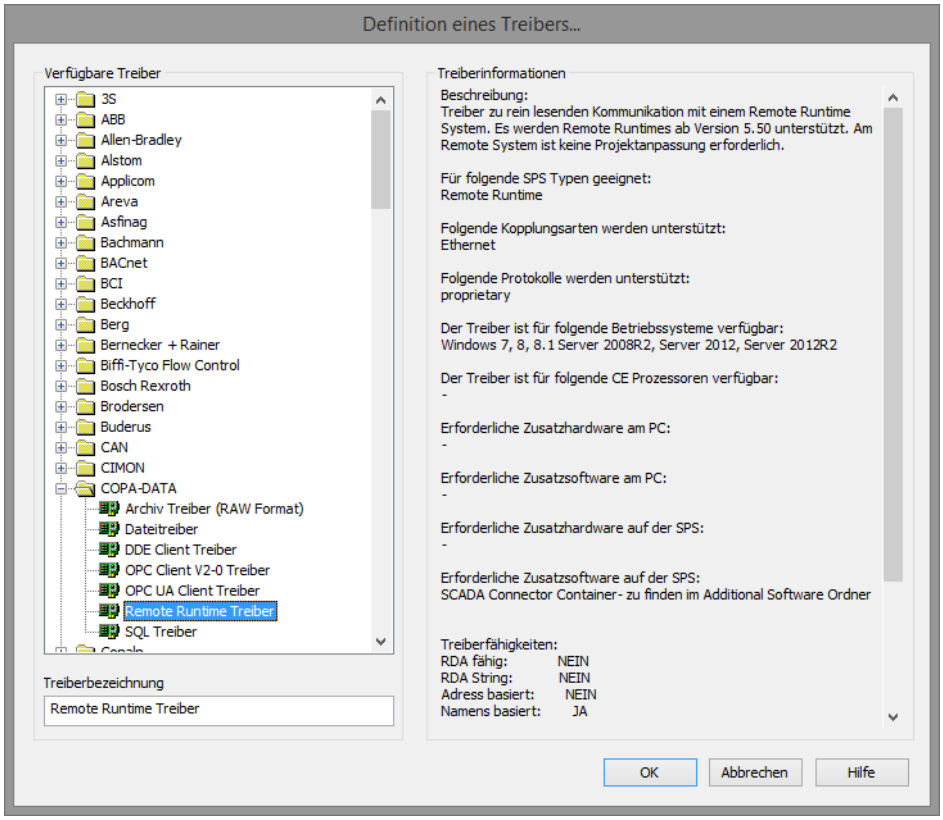
6 Konfiguration

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*

Parameter	Beschreibung
	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 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

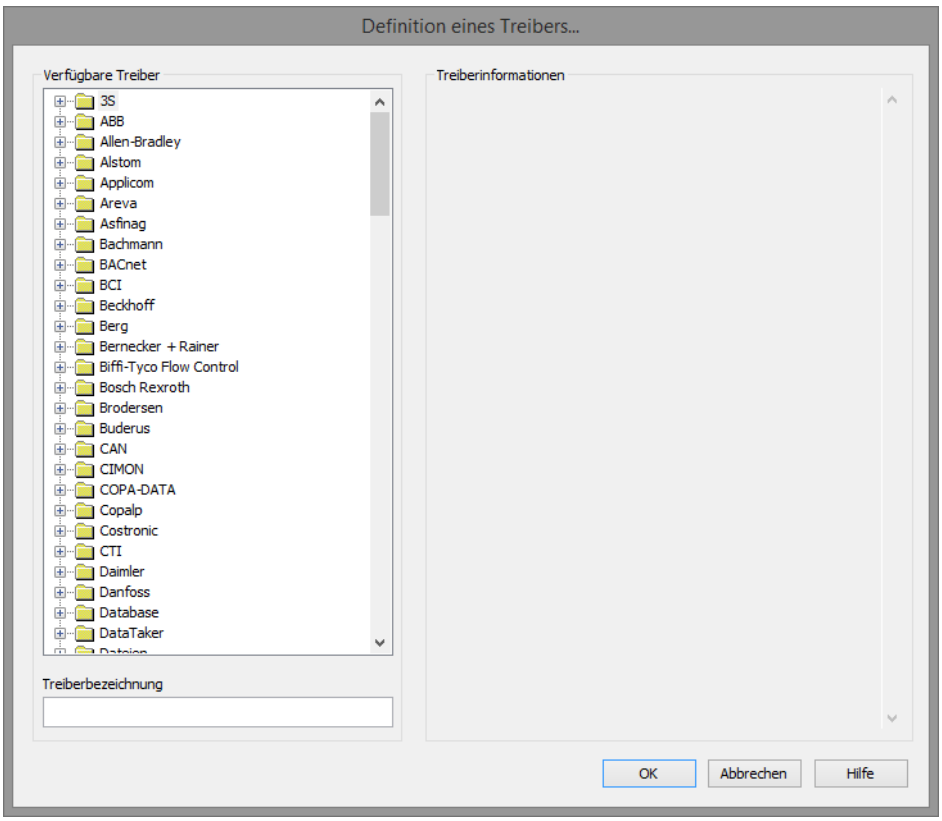
Die Inhalte dieses Dialogs sind in der Datei Treiber_[Sprachkürzel].xml gespeichert. Sie finden diese Datei im Ordner *C:\ProgramData\COPA-DATA\zenon[Versionsnummer]*.

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:

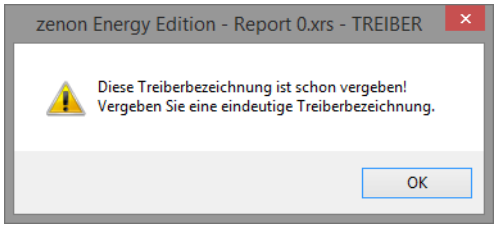
   - Die **Treiberbezeichnung** muss eindeutig sein.
     Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
     Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.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

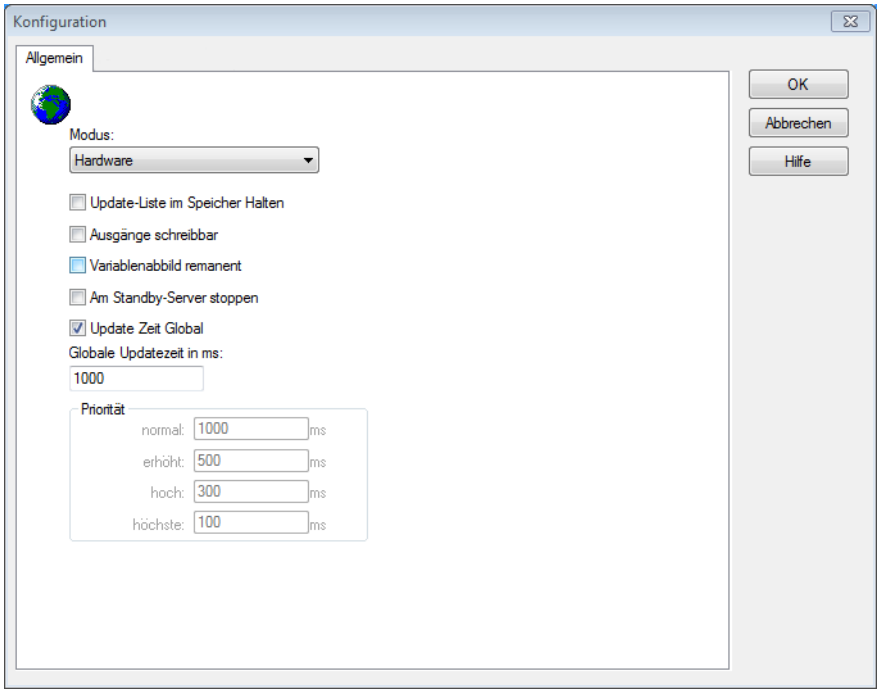
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

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.

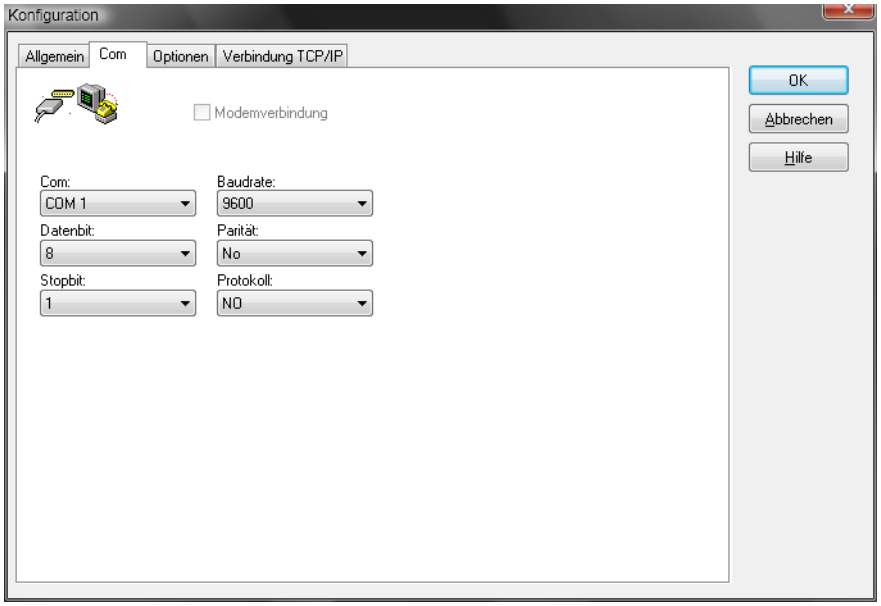
UPDATE ZEIT SPONTANE TREIBER

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 Com

In diesem Dialog konfigurieren Sie die Verbindungsparameter für die serielle Kommunikation zur SPS.



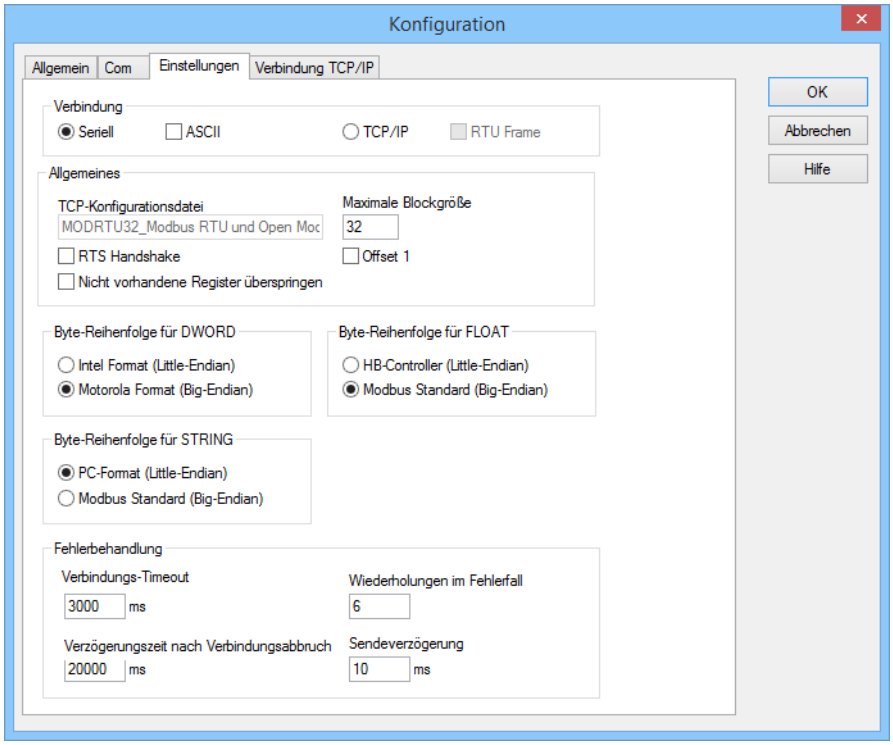
Parameter	Beschreibung
Com	Auswahl der seriellen Schnittstelle (Com-Port) am Rechner. Auswahl aus Dropdownliste: *COM 1* bis *COM 256* Default: *1*
Baudrate	Auswahl der Baudrate für die Kommunikation zur SPS. Die Baudrate muss an die Steuerung angepasst werden. Auswahl aus Dropdownliste: *110* bis *1152000* Default: *9600*
Datenbit	Anzahl der Datenbits (Datenwortlänge in Bit) für die Kommunikation zur SPS. Die Datenbitrate muss an die Steuerung angepasst werden. Auswahl aus Dropdownliste: *5, 6, 7, 8* Default: *8*
Stoppbit	Auswahl des Stoppbits für die Kommunikation zur SPS.

Parameter	Beschreibung
	Das Stoppbit muss an die Steuerung angepasst werden. Auswahl aus Dropdownliste: ▸ *1* ▸ *1.5* ▸ *2* Default: *1*
Parität	Auswahl der Parität für die Kommunikation zur SPS. Das Parität muss an die Steuerung angepasst werden. Anpassen an Steuerung. Auswahl aus Dropdownliste: ▸ *No* ▸ *Odd* ▸ *Even* Default: *No*
Protokoll	Auswahl des Protokolls für die Kommunikation zur SPS. Das Protokoll muss an die Steuerung angepasst werden. Auswahl aus Dropdownliste: ▸ *No* ▸ *Rts/CTS* Default: *No*

DIALOG BEENDEN

OK	**Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.** **Nur verfügbar, wenn keine Verbindung im Zustand *bearbeiten* ist.**
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

6.2.3 Einstellungen



VERBINDUNG

Parameter	Beschreibung
Verbindung	Definition, welche Anbindung verwendet wird.
Seriell	Seriell steht sowohl **Modbus RTU** als auch **Modbus ASCII** zur Auswahl.
ASCII	▸ *Aktiv:* Der Treiber kommuniziert mit Modbus ASCII Protokoll. *Inaktiv*: Der Treiber kommuniziert mit **Modbus RTU.**
TCP/IP	Benötigt eine Konfigurationsdatei. In dieser werden die TCP/IP Einstellungen gespeichert.
RTU Frame	*Aktiv:* Serielle Information wird 1:1 ins TCP-Protokoll übertragen.

ALLGEMEINES

Parameter	Beschreibung
Allgemeines	Allgemeine Einstellungen.
TCP-Konfigurationsdatei	Hier wird der Name der TCP Konfigurationsdatei festgelegt. Default: **MODRTU32_[Treiberbezeichnung].txt**
Maximale Blockgröße	Maximale Blockgröße eines Datentelegramms in WORDs. **Achtung zenon vor 5.50 SP6**: Blockgröße muss kleiner 64 sein. (1-125, Default 32)
RTS Handshake	Manche Steuerungen benötigen den RTS Handshake zur korrekten Kommunikation.
Offset 1	Der „Offset 1" wirkt sich auf die Adressierung aller Variablen aus. *Aktiv*: Der Treiber subtrahiert beim Senden die Variablenadressen (Coils, Register) jeweils um *1* und addiert *1* beim Empfang.
Nicht vorhandene Register überspringen	Daten werden vom Treiber in einem Block gelesen. Sind jedoch zwischen zwei nahe liegenden Adressbereichen keine Daten vorhanden, würde der Treiber versuchen, auf diese nicht vorhandenen Bereiche zuzugreifen. Diese Option verhindert dies.

BYTE-REIHENFOLGE FÜR DWORD

Parameter	Beschreibung
Byte-Reihenfolge für DWORD	Definiert die Reihenfolge von niederwertigem und höherwertigem Wort bei Doppelwort-Objekten (DINT/UDINT). Es kann zwischen Motorola (Big-Endian) und Intel (Little-Endian) gewählt werden.
Motorola Format (Big-Endian)	*Aktiv*: DWORD Ordering nach Motorola Format.
Intel Format (Little-Endian)	*Aktiv*: DWORD Ordering nach Intel Format.

Hinweis: Die Reihenfolge der Sortierung von niedrigeren und höheren Werten für Bytes innerhalb eines **WORD** entspricht immer dem Motorola-Format.

BYTE-REIHENFOLGE FÜR FLOAT

Parameter	Beschreibung
Byte-Reihenfolge für FLOAT	Definiert die Reihenfolge von niederwertigem und

Parameter	Beschreibung
	höherwertigem Wort bei FLOAT-Objekten (REAL). Es kann zwischen Modbus Standard (Big-Endian) und HB-Controller (Little-Endian) gewählt werden.
Modbus Standard (Big-Endian)	*Aktiv*: Floatordering nach Standard Modbus.
HB Controller Float (Little-Endian)	*Aktiv*: Floatordering nach HB Controller.

Hinweis: Die Reihenfolge der Sortierung von niedrigeren und höheren Werten für Bytes innerhalb eines **WORD** entspricht immer dem Motorola-Format.

BYTE-REIHENFOLGE FÜR STRING

Parameter	Beschreibung
Byte-Reihenfolge für STRING	Definiert Darstellung der Byte-Reihenfolge. **Hinweis:** Falls in der Dokumentation einer SPS eine Angabe enthalten ist, gilt meistens MSB-first = Motorola = Big-Endian = Modbus. Und: Intel = Little-Endian = PC. Das kann in manchen Dokumentationen aber unterschiedlich gehandhabt werden.
Modbus Standard (Big-Endian)	*Aktiv*: Darstellung nach Modbus Standard mit vertauschten Zeichen.
PC-Format (Little-Endian)	*Aktiv*: Darstellung im PC-Format mit aufsteigender Reihenfolge.

FEHLERBEHANDLUNG

Parameter	Beschreibung
Fehlerbehandlung	
Verbindungs-Timeout	Zeit in Millisekunden, die auf eine Antwort vom Slave gewartet wird. Kommt keine Antwort innerhalb dieser Zeit, wird ein Kommunikationsfehler ausgegeben. Nur bei serieller Verbindung einstellbar. **Hinweis:** Die **Kommunikations-Timeout** Zeit muss auf jeden Fall höher sein als die maximale Antwortzeit des ModbusRTU Slave. Es kann sonst zur Fehlinterpretation der Antworten kommen. Default: *3000* ms

Parameter	Beschreibung
Wiederholungen im Fehlerfall	Anzahl der Sendewiederholungen wenn nach der eingestellten **Kommunikations-Timeout** Zeit keine Antwort vom Slave erfolgt. *1*: ein Verbindungsversuch, keine Wiederholungen *0*: ständig Wiederholung Default: *6*
Verzögerungszeit nach Verbindungsabbruch	Zeit in Millisekunden die nach dem Auftreten eines Kommunikationsfehlers gewartet wird, bevor ein Wiederaufbau der Verbindung versucht wird. Default: *20000* ms
Sendeverzögerung	Zeitverzögerung in Millisekunden bei „send"-Aufträgen. Wirkt sich auf den gesamten Treiber aus. Ist nur bei serieller Verbindung auswählbar.

DIALOG BEENDEN

OK	**Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.** **Nur verfügbar, wenn keine Verbindung im Zustand *bearbeiten* ist.**
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

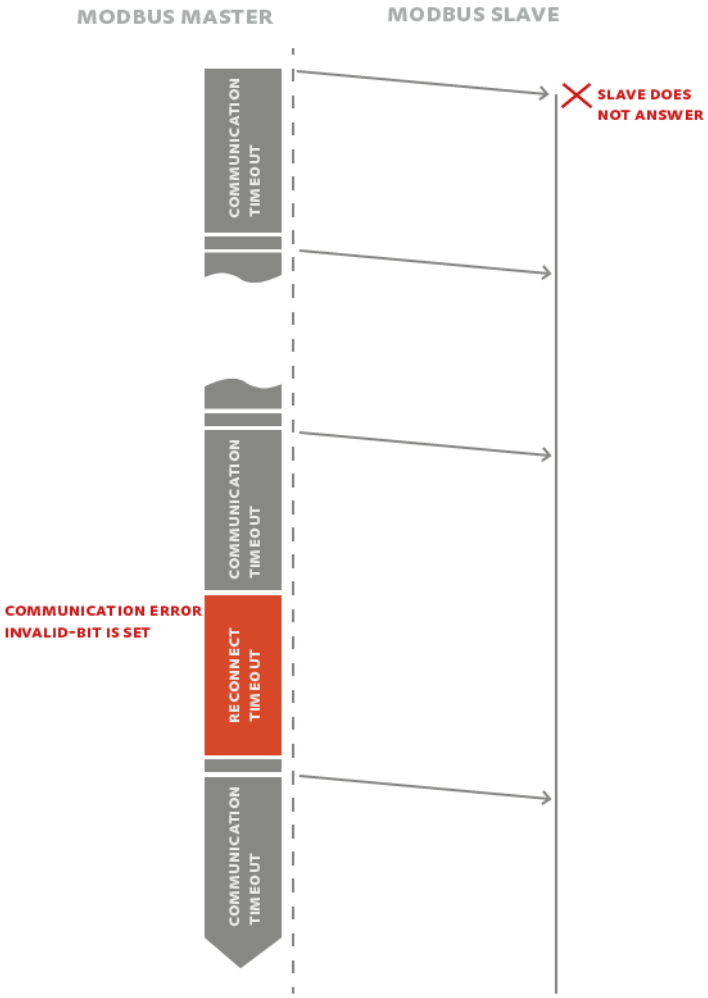
TIMEOUT VERHALTEN

Bei TCP Verbindung ist der Kommunikations-Timeout durch die systemabhängige Eigenschaft **Timeout [s]** vorgegeben.

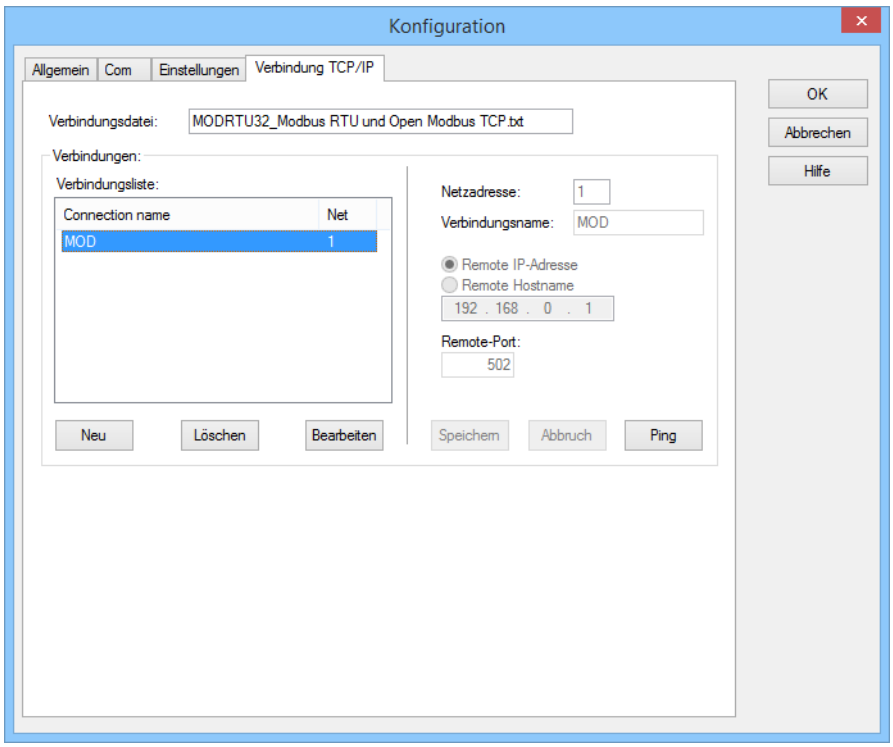
Nach dem Ablauf der Timeoutzeit versucht der Treiber bei Defaulteinstellungen bis zu 5 Mal, erneut die Anfragen zu senden. Sind diese 5 Versuche nicht erfolgreich, so wird für die Zeit des **Reconnect-Timeout** kein weiterer Verbindungsaufbau versucht.

Bei serieller Verbindung werden für das Timeout verhalten die Zeiten aus der Treiberkonfiguration verwendet. Vor dem Senden einer Anfrage wird die Sendeverzögerungszeit abgewartet. Nach Senden der Anfrage wird die für den Verbindungs-Timeout einegstellte Zeit auf eine Antwort gewartet. Kommt in dieser Zeit keine Antwort so wird bis zu 5 Mal noch versucht, die Anfrage erneut zu senden.

(Default 6 bedeutet 5 zusätzliche Versuche). Danach wird für die eingestellte Reconnect-Timeoutzeit kein weiterer Verbindungsversuch unternommen. Siehe dazu auch folgende Grafik.



6.2.4 Verbindung TCP/IP



Parameter	Beschreibung
Verbindungsdatei	Name der Datei, in der die Einstellungen für die einzelnen TCP/IP-Teilnehmer gespeichert werden. Nur Anzeige. Änderungen des Dateinamens erfolgen in der Registerkarte **Einstellungen** (auf Seite 19).

VERBINDUNGSLISTE

Einstellungen der Verbindungen.

Parameter	Beschreibung
Verbindungsliste	Liste aller definierten Verbindungen zu Steuerungen. ▸ *Connection name* Verbindungsname, wie in Verbindungsname konfiguriert. ▸ *Net:* Netzadresse der Vebindung wie in der Eigenschaft Netzadresse konfiguriert.
Neu	Erstellt eine neue Verbindung und schaltet die Eingabefelder im Bereich **Verbindungsparameter** frei.

Parameter	Beschreibung
Bearbeiten	Schaltet die Eingabefelder im Bereich **Verbindungsparameter** für eine gewählte Verbindung frei. Nicht aktiv, wenn keine Verbindung in der Verbindungsliste ausgewählt ist.
Löschen	Löscht eine ausgewählte Verbindung der Verbindungsliste. **Achtung:** Die Verbindung wird ohne Rückfrage gelöscht. Nicht aktiv, wenn keine Verbindung in der Verbindungsliste ausgewählt ist.

VERBINDUNGSPARAMETER

In diesem Bereich konfigurieren Sie die Einstellungen einer Verbindung.

Die Eingabe wird validiert. Im Fehlerfall wird ein entsprechender Warndialog angezeigt.

Parameter	Beschreibung
Netzadresse	Jeder Verbindung wird eine Netzadresse zugewiesen. Diese muss mit den Einstellungen in der Eigenschaft **Netzadresse** bei der Variablendefinition übereinstimmen. Default: *1* **Achtung:** Der Wert *0* ist reserviert für Broadcasts. Der Wert *255* ist für das Gateway reserviert. **Hinweis:** Bei MODBUS TCP-Kommunikation über ein Gateway (TCP/IP auf RS485) muss hier die Adresse der Zielstation (die sogenannte Modbus Unit ID oder SlaveAdresse) verwendet werden. Bei direkter MODBUS TCP-Verbindung wird das **Unit ID** Adressfeld von manchen Geräten ignoriert, da die TCP/IP-Adresse der Steuerung prinzipiell ausreichend ist. In diesem Fall kann hier ein beliebiger Wert eingestellt werden.
Verbindungsname	Name der Verbindung, wie er in der Verbindungsliste angezeigt wird. Default: *Defaultname*
Remote IP-Adresse	*Aktiv*: Die Steuerung wird über die IP-Adresse angesprochen. Eingabe in das Feld **Eingabe Adresse**.

Parameter	Beschreibung
Remote Hostname	*Aktiv*: Die Steuerung wird über den Hostnamen angesprochen. Eingabe in das Feld **Eingabe Adresse**.
Eingabe Adresse	Eingabe der Adresse der Steuerung. Abhängig von den Einstellungen des übergeordneten Optionsfeldes erfolgt die Eingabe als: ▸ IP-Adresse Default: *192.168.0.1* ▸ Hostname Default: *localhost*
Remote-Port	Eingabe der Port-Adresse der Steuerung. Details finden Sie im Handbuch Ihrer Steuerung. Standardport: *502*
Ping	Sendet einen Ping zur IP-Adresse, die für diese Verbindung konfiguriert wurde. Ermöglicht das Testen der Verbindung zum Gerät. Wird der Ping negativ abgeschlossen, prüfen Sie die IP-Adresse und ob das Gerät online ist.
Speichern	Übernimmt alle Änderungen für bearbeitete Verbindung und schließt Editiermöglichkeit.
Abbruch	Verwirft alle Änderungen für bearbeitete Verbindung und schließt Editiermöglichkeit.

DIALOG BEENDEN

OK	**Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.** **Nur verfügbar, wenn keine Verbindung im Zustand *bearbeiten* ist.**
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

NEUE VERBINDUNG ANLEGEN

1. Klicken Sie auf die Schaltfläche **Neu**.

2. Tragen Sie die Verbindungsdetails ein.
3. Klicken Sie auf **Speichern.**

VERBINDUNG BEARBEITEN

1. Wählen Sie in der Verbindungsliste die gewünschte Verbindung.
2. Klicken Sie auf die Schaltfläche **Bearbeiten.**
3. Ändern Sie die Verbindungsparameter.
4. Schließen Sie mit **Speichern** ab.

VERBINDUNG LÖSCHEN

1. Wählen Sie in der Verbindungsliste die gewünschte Verbindung.
2. Klicken Sie auf die Schaltfläche **Löschen.**
3. Die Verbindung wird aus der Liste gelöscht.

7 Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

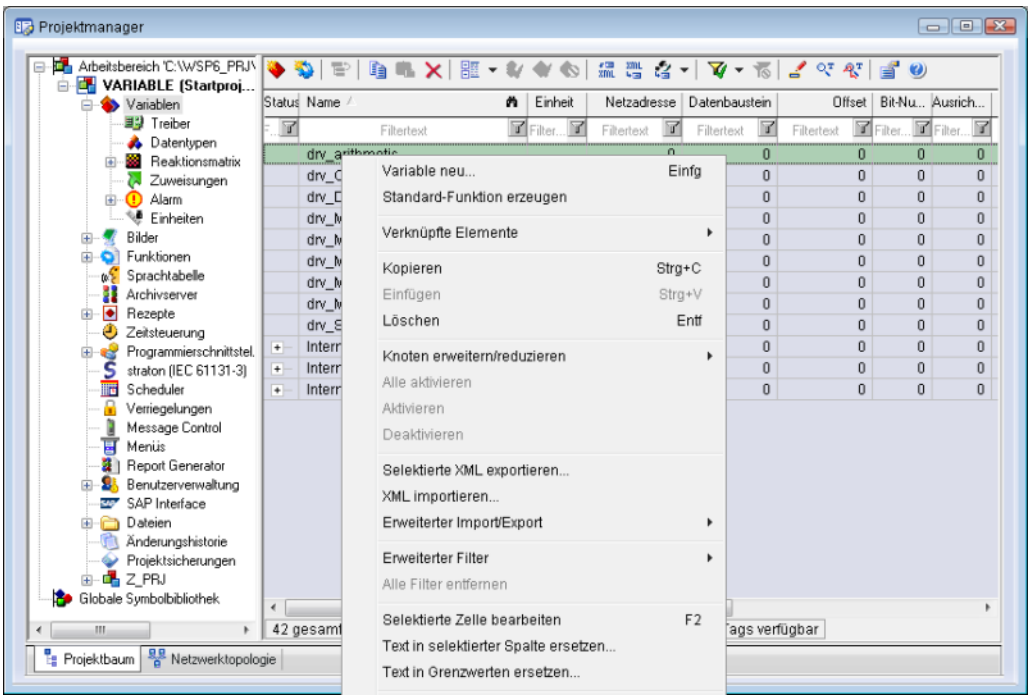
Variablen können angelegt werden:

- 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:

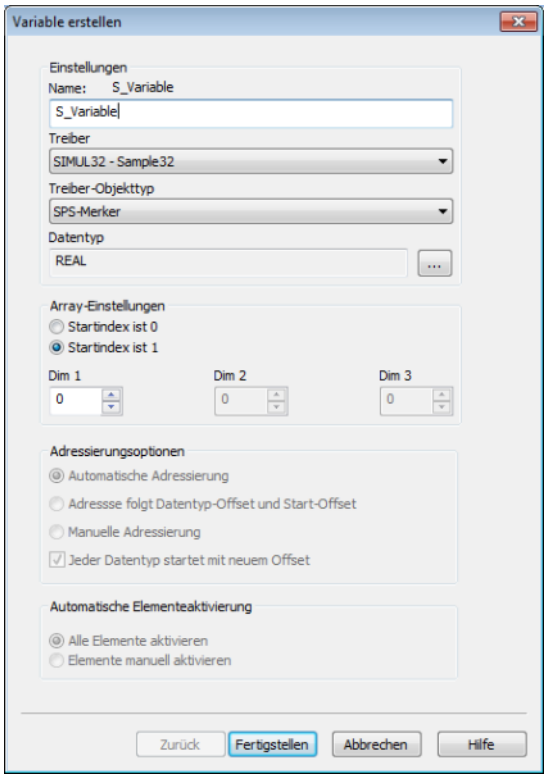
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



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

Eigenschaft	Beschreibung
Name	Frei vergebbarer Name. **Achtung:** Je zenon Projekt muss der Name eindeutig sein.
Kennung	Frei vergebbare Kennung. Z. B. für **Betriebsmittelkennung**, Kommentar usw.
Netzadresse	Busadresse (Slave-Adresse) und Netzadresse der Variable. Diese Adresse bezieht sich für serielle Kommunikation auf die Modbusadresse der Steuerung - die Adresse des Empfängers. Bei Kommunikation über Ethernet bezieht sich die Netzadresse sowohl auf die Verbindungsprojektierung im Treiber als auch auf die Modbus Unit ID. Damit wird ausgewählt auf welcher Steuerung/Verbindung sich die Variable befindet. Wird über ein Gateway (z.B. TCP/IP auf RS485) mit den einzelnen Modbusstationen kommuniziert, so ist für jede Modbusstation eine eigene TCP/IP-Verbindung anzulegen. Als IP-Adresse ist die Adresse des Gateways anzugeben, als Netzadresse der Verbindung und in den Variablen die Adresse der Modbusstation - Modbus Unit ID. Höchste mögliche Modbus Adresse lt. Protokollspezifikation ist *247*. Adresse *0* wird für Broadcast Messages verwendet (nur schreibend).
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Offset der Variablen. Entspricht der Speicheradresse der Variablen in der Steuerung. Einstellbar von *0* bis *4294967295*.
Ausrichtung	Der Treiber adressiert Wortbasierend (16Bit), wird nur ein Byte gelesen, so kann hier eingestellt werden ob das HiByte oder das LowByte angesprochen werden soll.

Eigenschaft	Beschreibung
Bitnummer	Nummer des Bits innerhalb des eingestellten Offsets. Mögliche Eingabe: *0* bis *65535*. Funktionierender Bereich [0-15]
Stringlänge	Nur verfügbar bei String-Variablen. Maximale Anzahl von Zeichen, die die Variable aufnehmen kann.
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.

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:

Treiber-objekttyp	Kanaltyp	Lesen: Modbus Funktion (Code hex/dec)	Schreiben: Modbus Funktion (Code hex/dec)	Unterstützte Datentypen	Kommentar
Alarm Stack	97	0x03/3	N/A	*STRING*	Spezielle Events aus Secheron Steuerung auslesen. Adressierung: ▸ Event-Typ = **Datenbaustein** ▸ Subtyp = **Offset**.
Analog Input	10	0x04/4	N/A	*REAL, LREAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING*	Class 1 - input register. Lineare Adressierung wie *Holding Register*.
Byte aligned Coil	69	0x01/1	0x0F/15	*BOOL*	Class 1 - coil. Lineare Adressierung einstufig über **Offset.**
Coil	65	0x01/1	0x05/5	*BOOL*	Class 1 - coil. Lineare Adressierung einstufig über **Offset.**
Device Status	24	0x11/17	N/A	*BOOL, USINT*	SPS-spezifisch.
Device Identification	68	0x2B/43	N/A	*STRING*	SPS-spezifisch.
Input Status	66	0x02/2	N/A	*BOOL*	Class 1 - input discretes. Lineare Adressierung einstufig über **Offset.**
Holding Register	8	0x03/3	0x10/16	*REAL, LREAL, BOOL, DINT, UDINT,*	Class 0 - multiple register. Lineare Adressierung:

Treiber-objekttyp	Kanaltyp	Lesen: Modbus Funktion (Code hex/dec)	Schreiben: Modbus Funktion (Code hex/dec)	Unterstützte Datentypen	Kommentar
				USINT, INT, UINT, SINT, STRING, WSTRING	▸ Bool (1 Bit): einstufig über **Offset** und **Bit-Nummer** ▸ Byte (8 Bits): einstufig über **Offset** und **Ausrichtung** ▸ Wort (16 Bits) Doppel-Wort (32 Bits) Float (32 Bits) String (n*Byte): einstufig über **Offset** ▸ WString (n*Word): einstufig über **Offset**
Holding Register 32 Bit	70	0x03/3	0x10/16	*REAL*	Pro Offset werden 32 Bit (4 Byte) gelesen.
Holding Register 64 Bit	71	0x03/3	0x10/16	*LREAL*	Pro Offset werden 64 Bit (8 Byte) gelesen.
Preset Single Register (FC 6)	67	N/A	0x06/6	*UINT*	Clase 1 - single register.
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).

Treiber-objekttyp	Kanaltyp	Lesen: Modbus Funktion (Code hex/dec)	Schreiben: Modbus Funktion (Code hex/dec)	Unterstützte Datentypen	Kommentar
					Hinweis: Die Adressierung und das Verhalten ist bei den meisten zenon Treibern gleich. Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 44).

Legende:

X: wird unterstützt

--: wird nicht unterstützt

MODBUS FUNCTIONCODES

Functioncode hex/dec	Modbus Bezeichner	Kommentar
0x01/1	Read coils	This function code is used to read from 1 to 2000 contiguous status of coils (bits) in a remote device
0x02/2	Read inputs discretes	This function code is used to read from 1 to 2000 contiguous status of discrete inputs (bits) in a remote device
0x03/3	Read multiple registers	This function code is used to read a block of contiguous holding registers (1 to 125 words) in a remote device.
0x04/4	Read input registers	This function code is used to read a block of contiguous input registers (1 to 125 words) in a remote device.
0x05/5	Write coil	This function code is used to write a single output (one bit) to either ON or OFF in a remote device.

Functionco de hex/dec	Modbus Bezeichner	Kommentar
0x06/6	Write single register	This function code is used to write a single (one word) holding register in a remote device.
0x10/16	Write multiple registers	This function code is used to write a block of contiguous holding registers (1 to approx. 120 words) in a remote device.
0x11/17	Report Slave ID	This function code is used to read the description of the type, the current status, and other information specific to a remote device.
0x14/20	Read File Record	This function code is used to perform a file record read. All Request Data Lengths are provided in terms of number of bytes and all Record Lengths are provided in terms of registers. **Note:** This Functioncode is supported by the MODBUS_ENERGY driver only.
0x15/21	Write File Record	This function code is used to perform a file record write. All Request Data Lengths are provided in terms of number of bytes and all Record Lengths are provided in terms of the number of 16-bit words. **Note:** This Functioncode is supported by the MODBUS_ENERGY driver only.
0x2B/43	Read Device Identification	This function code allows reading the identification and additional information relative to the physical and functional description of a remote device.

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.

BEISPIEL FÜR ALLE MÖGLICHEN ZENON DATENTYPEN:

Steuerung	zenon	Datenart
BOOL	BOOL	8
BYTE	SINT	10
BYTE	USINT	9
WORD	INT	1
WORD	UINT	2
DWORD	DINT	3
DWORD	UDINT	4
FLOAT	REAL	5
FLOAT	LREAL	6
STRING	STRING	12
WSTRING	WSTRING	21

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.

Bezeichnung	Typ	Feldgröße	Bemerkung
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)
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)

Bezeichnung	Typ	Feldgröße	Bemerkung
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, 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

Bezeichnung	Typ	Feldgröße	Bemerkung
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

Bezeichnung	Typ	Feldgröße	Bemerkung
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
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.

7.5 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.

Name aus Import	Typ	Offset	Erklärung
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 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

Name aus Import	Typ	Offset	Erklärung
			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 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.

Name aus Import	Typ	Offset	Erklärung
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. 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.

Name aus Import	Typ	Offset	Erklärung
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

Name aus Import	Typ	Offset	Erklärung
			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:

BLOCKWRITE

Zum effizienteren Absetzen von vielen Sollwerten (z. B. Rezepte) von Variablen des Treiberobjekttyps *Holding Register* kann die Eigenschaft Blockwrite aktiviert werden. Damit werden Variablen die im Steuerungsspeicher hintereinander liegen mit einem Schreibtelegramm beschrieben. Bei größeren Bereichen wird das Schreiben in wenige Telegramme zusammengefasst, anstatt jede Variable einzeln zu beschreiben.

AKTIVIEREN DES BLOCKWRITE

Folgender Eintrag muss in die **project.ini** eingetragen werden:

[MODRTU32]

BLOCKWRITE=1

TRANSACTION IDENTIFICATION

Der Treiber erhöht den Transaction Identifier bei jedem verschickten Paket und erwartet vom Slave eine Antwort mit dem gleichen Transaction Identifier oder *0*.

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.

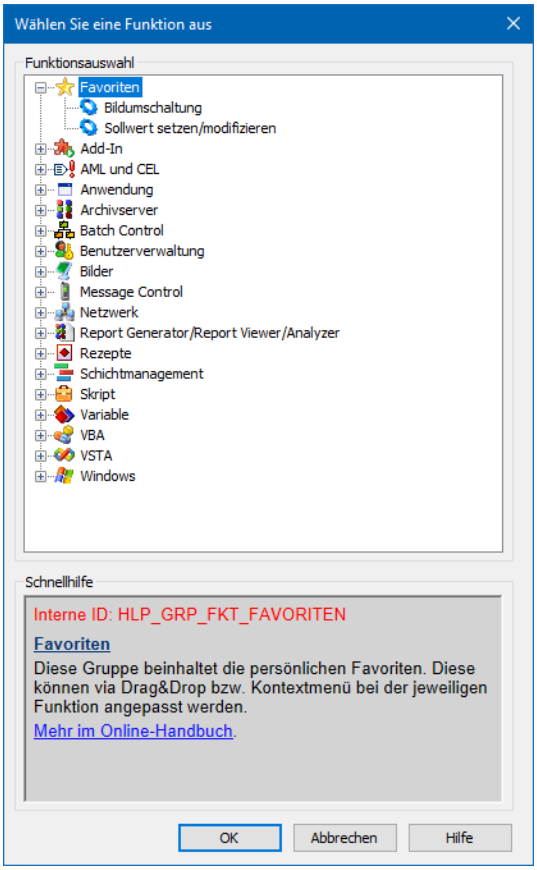
> **Achtung**
>
> Die zenon Funktion **Treiberkommandos** ist nicht ident mit den Treiberkommandos, die bei Energy-Treibern in der Runtime ausgeführt werden können!

PROJEKTIERUNG DER FUNKTION

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

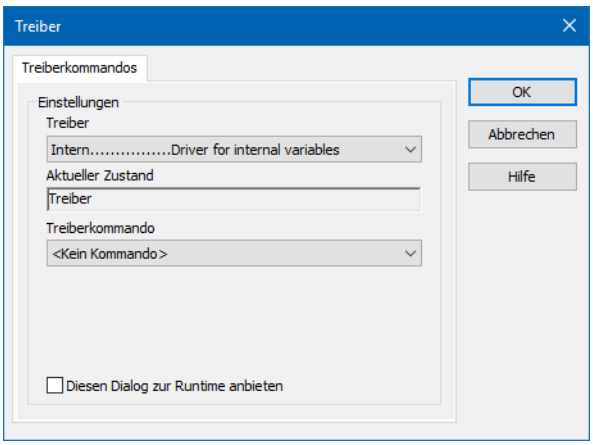
1. Legen Sie im zenon Editor eine neue Funktion an.

   Der Dialog zur Auswahl einer Funktion wird geöffnet.



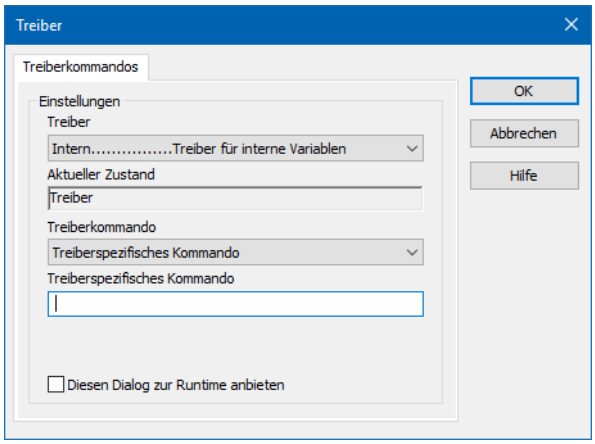
2. Navigieren Sie zum Knoten **Variable.**
3. Wählen Sie den Eintrag **Treiberkommandos**.

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

Folgende Fehlercodes gelten NUR für die Fehlermeldungen

CallSPSread FAILED ! ERROR:' und

'CallSPSwrite FAILED ! ERROR:'

-1	Unzulässige Funktion
-2	Unzulässige MW-Adresse
-3	Unzulässiges Datum
-4	Fehler im Slave
-7	Slave kann die Funktion nicht ausführen
-8	Speicherparitätsfehler im Slave
-10	Timeout bei der Kommunkation mit Slave
-11	Anfrage zum Lesen fehlgeschlagen, da keine Kommunikation zur Steuerung möglich.

Die Felder Para1 und Para2 enthalten zusätzliche Informationen über die Variable die nicht gelesen werden kann:

Die Zahlen beziehen sich auf die interne ID

Para1: Kanaltyp (z.B: 8=Holding Register)

8: Holding Register

10 Input Register

24 Status

65 COIL

66 Input Status

67 Preset Single Register

Para2: Datentyp (z.B: 2= Word, UINT)

1: INT

2: UINT

3: DINT

4: UDINT

5: REAL

8: BOOL

9: SINT

10: USINT

10.3 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. ▸ Der PC ist nicht am Netzwerk angeschlossen. ▸ 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. ▸ Ist die Firewall aktiviert? Als Standard wird Port 502 zur Kommunikation verwendet, zu den Ausnahmen hinzufügen. Bei individueller Portnummer entsprechend diese eintragen.
	Die Steuerung kann per Ping erreicht werden?	▸ Die Kommunikationsparameter sind nicht korrekt eingestellt? ▸ Der Port muss entsprechend der konfiguration des Steuerung (Modbus Slave) eingestellt sein. ▸ Die Netzadresse in der Adressierung der Variable stimmt mit der Netzadresse der Verbindung im Treiber nicht überein. Achtung: Netzadresse 0 darf nicht verwendet werden. Adresse 0 ist in Modbus als Broadcast Adresse reserviert. ▸ Die Treiberkonfigurations Datei wurde nicht auf den Zielrechner übertragen?
	Die Steuerung ist seriell angeschlossen	▸ Die Steuerung ist nicht an die Stromversorgung und/oder Bussystem angeschlossen. ▸ Das serielle Kabel ist nicht an die korrekte Schnittstelle (COM1...64) angeschlossen,

Problem	Diagnose	Ursache
		oder die schnittstelle wurde falsch eingestellt. ▸ Die serielle Schnittstelle ist durch eine weitere Anwendung blockiert. ▸ Die Netzadresse in der Adressierung der Variable stimmt mit der Modbusadresse (Deviceadresse) nicht überein. ▸ Das Kabel ist falsch belegt, oder defekt.
Es können bestimmte Werte von der Steuerung nicht gelesen oder geschrieben werden.	Wurde mit dem Diagnosis Viewer bereits analysiert, welche Fehler sind aufgetreten? Siehe Kapitel Analysetool (auf Seite 56)	▸ Siehe Kapitel: Fehlernummern (auf Seite 57) ▸ Sind die Variablen korrekt adressiert? ▸ 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? Siehe Kapitel Analysetool (auf Seite 56)	▸ Sind die Variablen korrekt adressiert? ▸ Sind die korrekten Datentypen verwendet? ▸ Ist die Wertberechnung korrekt?