zenon Treiber Handbuch

MODBUS_ENERGY

v.7.60

©2017 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.

Inhalt

1. Willkommen bei der COPA-DATA Hilfe 5

2. MODBUS_ENERGY 5

3. MODBUS_ENERGY - Datenblatt 6

4. Treiber-Historie 8

5. Voraussetzungen 8

5.1 Steuerung 9

6. Konfiguration 9

6.1 Anlegen eines Treibers 10

6.2 Einstellungen im Treiberdialog 13

6.2.1 Allgemein 14

6.2.2 Einstellungen 18

6.2.3 Verbindungen 21

7. Variablen anlegen 25

7.1 Variablen im Editor anlegen 25

7.2 Adressierung 27

7.3 Treiberobjekte und Datentypen 29

7.3.1 Treiberobjekte 29

7.3.2 Zuordnung der Datentypen 35

7.4 Variablen anlegen durch Import 36

7.4.1 XML Import 37

7.4.2 DBF Import/Export 38

7.5 Kommunikationsdetails (Treibervariablen) 44

8. Treiberspezifische Funktionen 49

8.1 ICE NPx800 53

8.2 GE Multilin UR-series 56

8.3 COSTRONIC DFB 57

9. Treiberkommandos 59

10. Fehleranalyse 60

10.1 Analysetool 60

10.2 Checkliste 62

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 bitte per E-Mail an documentation@copadata.com (mailto: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 (mailto:support@copadata.com) erreichen.

LIZENZEN UND MODULE

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

2. MODBUS_ENERGY

Der Open Modbus TCP/IP Treiber unterstützt für Energy-Projekte Sequence of Events und außerdem (auch für nicht-Energy Projekte) Kommunikation über ein TCP/IP Gateway mit mehreren Slaves am seriellen Bus.

Bei der Verwendung eines Modbus Gateways kann der Treiber so konfiguriert werden, dass der Treiber auch bei mehreren Geräten (Slaves) hinter dem Gateway nur eine TCP-Verbindung zum Gateway aufbaut.

Der Treiber unterstützt das Auslesen des Ereignis-Puffers (Events) von:

- AREVA MiCOM P125/P126/P127
- COSTRONIC DFB: unterstützt das Lesen von Events von der Schneider Modicon TSX Premium SPS mit dem von der Costronic SA (www.costronic.ch (http://www.costronic.ch)) erstellten Funktionsblock (DFB) „lynxPileHorodatageV3a".
- GE Multilin F650 und UR-Serie Steuerungen
- IEC NPx800
- Schneider SEPAM

3. MODBUS_ENERGY - Datenblatt

Allgemein:	
Treiberdateiname	MODBUS_ENERGY.exe
Treiberbezeichnung	Modbus Energy Treiber
Steuerungs-Typen	Alle Steuerungen und Gateways mit Open Modbus TCP/IP Unterstützung. Sequence of Events nur für AREVA MiCOM P126/P127, GE Multilin F650, GE Multilin UR-series, ICE NPI 800, ICE NPID 800, Schneider MODICON TSX (Premium,M340,Quantum) mit Funktionsblock (DFB) von Costronic und Schneider Sepam Schutz Relais.
Steuerungs-Hersteller	ABB; GE Automation&Controls; Modbus RTU; Mondial; Schiele; Telemecanique; Schneider; Wago; SE Elektronic; Areva; GE Multilin; ICE; Costronic;

Treiber unterstützt:	
Protokoll	Modbus RTU over TCP;
Adressierung: Adress-basiert	X
Adressierung: Namens-basiert	--
Kommunikation spontan	X

Kommunikation pollend	X
Online Browsing	--
Offline Browsing	--
Echtzeitfähig	X
Blockwrite	X
Modemfähig	--
Serielles Logging	--
RDA numerisch	X
RDA String	--
Hysterese	X
erweiterte API	--
Unterstützung von Statusbit **WR-SUC**	X
alternative IP-Adresse	X

Voraussetzungen:	
Hardware PC	--
Software PC	--
Hardware Steuerung	--
Software Steuerung	Für Schneider MODICON TSX Premium: Funktionsblock (DFB) "cosZenonPileHorodatageV2c" von Costronic
Benötigt v-dll	--

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
26.03.09	500	Treiberdokumentation wurde neu erstellt der erste Freigabetest wurde durchgeführt (GE Multilin F650 und Areva MiCOM P127)
14.10.09	900	COSTRONIC DFB und IEC NPx800 hinzugefügt.
07.01.10	1100	GE Multilin UR-Serie hinzugefügt
09.06.10	1699	Schneider SEPAM hinzugefügt
15.01.2016	25506	Neuer Treiber Objekttyp `File record`
16.12.2016	33762	Direkte Redundanzumschaltung bei Ausfall (FS 37567)

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 Steuerung

Für den Einsatz des Modbus Energy Treibers gelten folgende Bedingungen:

- Die Steuerung muss das Open Modbus TCP/IP Protokoll in der Konformitätsklasse 0 unterstützen.
- Das Auslesen des Event-Buffers ist nur auf den im Kapitel MODBUS_ENERGY (auf Seite 5) genannten Steuerungen möglich.
- Die Steuerungen sollte, auf UTC-Zeit eingestellt werden. Der Treiber geht davon aus, dass alle kommenden Zeitstempel in der UTC-Zeit gegeben sind.

AREVA MICOM P125/P126/P127

Das Zeitformat in Areva MiCOM P12/P125/P127 Register 012Fh soll auf "Internes Format" (nicht auf "IEC Format") eingestellt sein.

COSTRONIC DFB

Der Block soll vom Task „FAST" ausgeführt werden und wie folgt Konfiguriert werden.

- **ControleVie** und **IndiceEchange** und **balEvenements** müssen hintereinander liegen, z. B. %MW900 und %MW901 und %MW902:56
- **balEvenements** muss 56 Register groß sein
- **nombreMaxEvenementBal** muss zwischen `1` und `7` liegen (Anzahl der Events in der Mailbox)
- An **Pile** wird der Stack zum Zwischenspeichern der Events verbunden
- **tableEvenments** enthält den Status aller Events (1 Bit pro Event)
- **numeroPremierEvenement** muss `0` sein.

6. Konfiguration

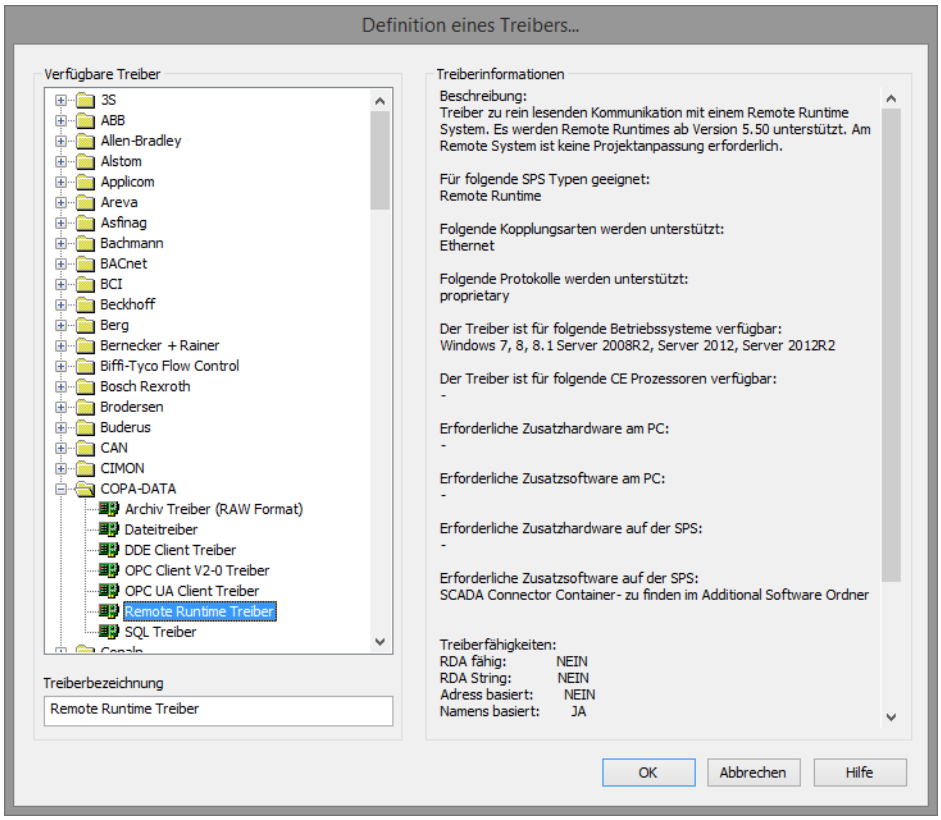
In diesem Kapitel lesen Sie, wie Sie den Treiber im Projekt anlegen und welche Einstellungen beim Treiber möglich sind.

Info

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

6.1 Anlegen eines Treibers

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



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

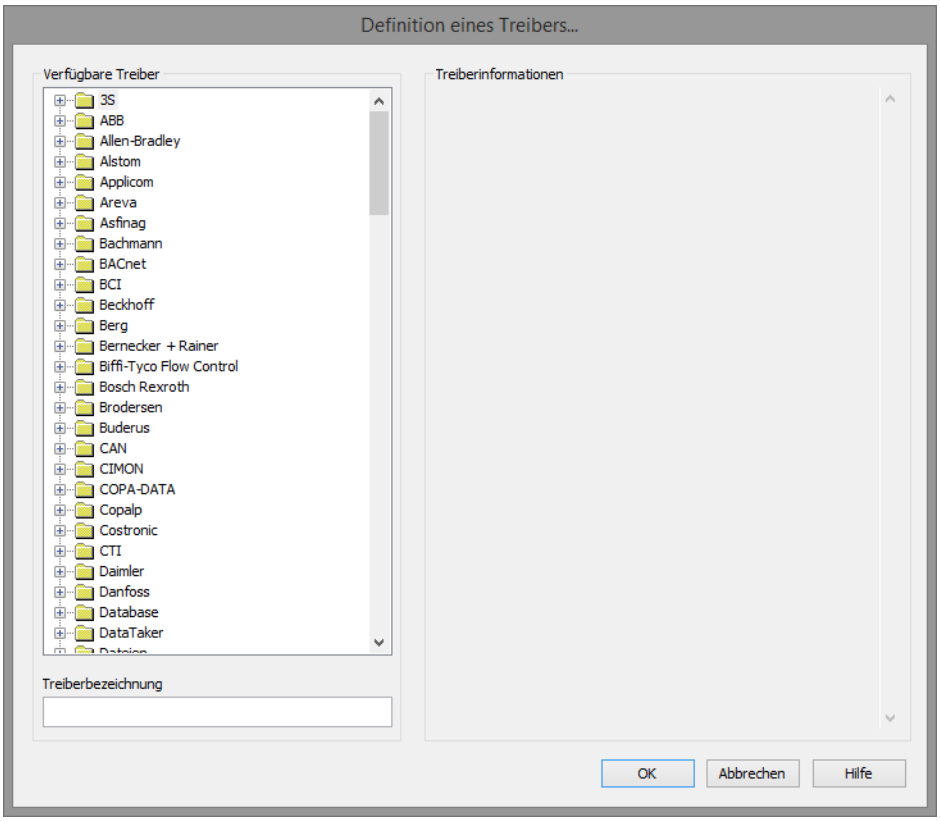
TREIBER NEU ANLEGEN

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:

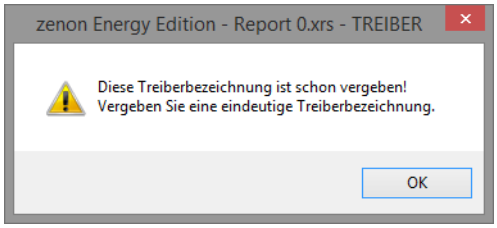
- Die **Treiberbezeichnung** muss eindeutig sein.
  Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
  Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.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



<CD_PRODUCNTAME> PROJEKT

Bei neu angelegten Projekten werden die folgenden Treiber automatisch angelegt:

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

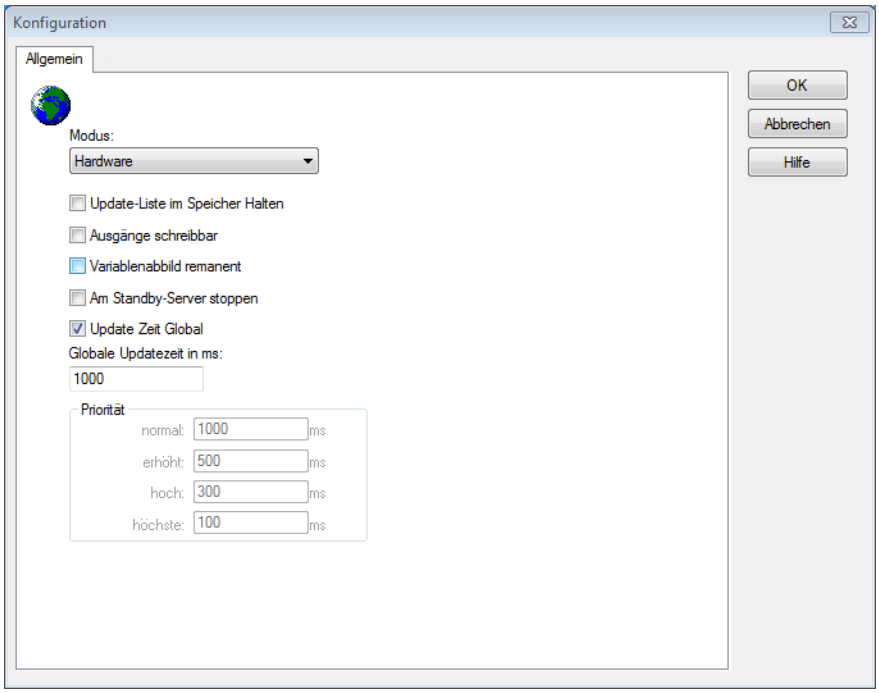
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

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. 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 der Status S_MERKER_1(0) bis S_MERKER8(7), REVISION(9), AUS(20) oder ERSATZWERT(27) aktiv ist

	Das Variablenabbild wird immer gespeichert wenn: ▸ die Variable vom Objekttyp **Kommunikationsdetails** 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. **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	`Aktiv`: Die eingestellte **Globale Update Zeit** in `ms` wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität wird nicht verwendet. `Inaktiv`: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet.
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. Zum Beispiel unterstützen spontan kommunizierende Treiber dies nicht.

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 ZYKLISCHE TREIBER

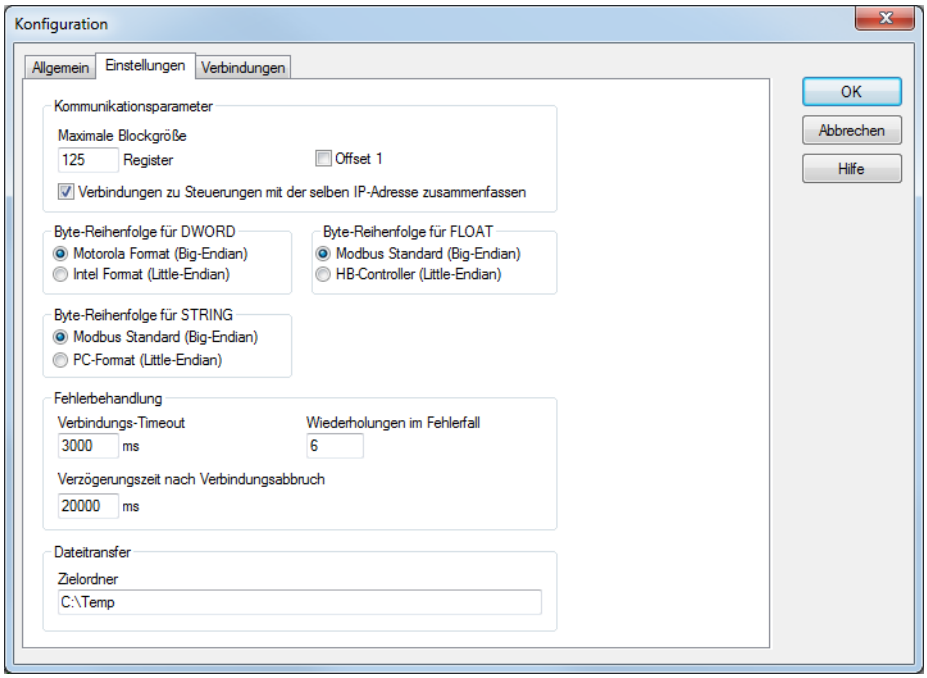
Für zyklische Treiber gilt:

Beim **Sollwert Setzen**, **Advisen** von Variablen und bei **Requests** wird sofort ein Lesezyklus für alle Treiber ausgelöst - unabhängig von der eingestellten Update Zeit. Damit wird sicher gestellt, dass der Wert nach dem Schreiben in der Visualisierung sofort zur Verfügung steht. Update-Zeiten können damit für zyklische Treiber kürzer ausfallen als eingestellt.

6.2.2 Einstellungen

Die allgemeinen Einstellungen für alle Open Modbus TCP/IP Verbindungen konfigurieren Sie in der Registerkarte **Einstellungen**. Dazu:

- klicken Sie auf die Eigenschaft **Konfiguration** in der Gruppe **Allgemein**
- der Dialog für die Konfiguration des Treibers öffnet sich
- wählen Sie die Registerkarte Einstellungen



KOMMUNIKATIONSPARAMETER

Parameter	Beschreibung
Maximale Blockgröße	Definiert die maximale Anzahl von Registern, die über ein Datentelegramm abgefragt bzw. geschrieben werden können. Der eingestellte Wert gilt für Schreib- und Leseanfragen. Wird ein Wert größer `100` eingestellt, werden für Schreibanfragen maximal 100 Register geblockt. Möglicher Wertebereich: `1` - `125`.
Offset 1	`Aktiv`: Der Treiber subtrahiert beim Senden die Variablenadressen (Coils, Register) jeweils um `1` und addiert `1` beim Empfang. Wirkt sich auf die Adressierung aller Variablen aus.
Verbindungen zu Steuerungen mit der selben IP-Adresse zusammenfassen	`Aktiv`: Verbindungen mit gleicher IP-Adresse und gleicher Portnummer werden zusammengefasst. Soll zu einem Gateway nur eine einzige TCP Verbindung aufgebaut werden, aktivieren sie diese Option.

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 (Littel-Endian) gewählt werden.
Motorola Format (Big-Endian	`Aktiv`: DWORD Ordering nach Motorola Format.
Intel Format (Little-Endian)	`Aktiv`: DWORD Ordering nach Intel Format.

BYTE-REIHENFOLGE FLOAT

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

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. ▶ Intel = Little-Endian = PC. Das kann in manchen Dokumentationen aber unterschiedlich gehandhabt werden.
Modbus Standard (Big-Endian)	Darstellung nach Modbus Standard mit vertauschten Zeichen.
PC-Format (Little-Endian)	Darstellung nach Modbus Standard mit vertauschten Zeichen.

FEHLERBEHANDLUNG

Diese Einstellungen sind relevant für eine Redundanzumschaltung bei Ausfall einer Verbindung.

Parameter	Beschreibung
Verbindungs-Timeout	Zeit in Millisekunden, die auf die Antwort eines Slaves gewartet wird. Kommt keine Antwort innerhalb dieser Zeit, wird ein Kommunikationsfehler ausgegeben. Default: `3000 ms`
Wiederholungen im Fehlerfall	Anzahl der Sendewiederholungen wenn nach eingestellter 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 Auftreten eines Kommunikationsfehlers gewartet wird, bevor ein erneuter Verbindungsaufbau versucht wird. Default: `20000 ms`
Zielordner	Ordner für Dateitransfer.

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.

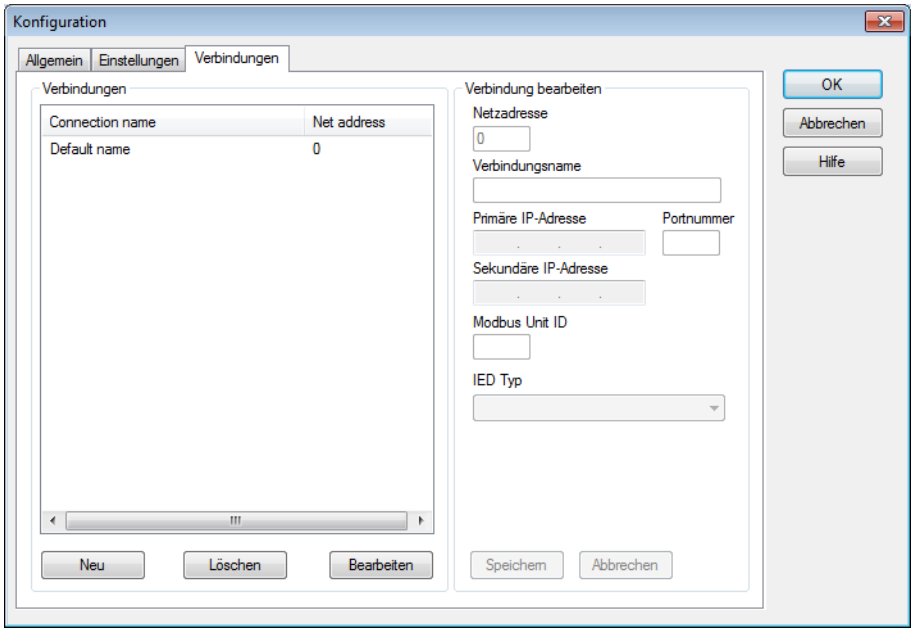
Info

Sequents of Events: Ist eine Variable für eine Verbindung im Eventbereich angelegt, wird SOE verwendet, sonst nicht.

6.2.3 Verbindungen

Die Verbindungen über Open Modbus TCP/IP zu den Steuerungen konfigurieren Sie in der Registerkarte **Verbindungen**. Dazu:

- klicken Sie auf die Eigenschaft **Konfiguration** in der Gruppe **Allgemein**
- der Dialog für die Konfiguration des Treibers öffnet sich
- wählen Sie die Registerkarte **Verbindungen**



Parameter	Beschreibung
Verbindungsliste	Zeigt die Verbindungsnamen mit der dazugehörigen Hardware-Adresse an. Um die Verbindungsparameter eines Namens anzuzeigen, klicken Sie mit der Maus auf den betreffenden Verbindungsnamen.
Netzadresse	Die Netzadresse identifiziert die Verbindung. Jede Verbindung muss daher eine eindeutige Netzadresse haben. Variablen werden einer Verbindung über die Netzadresse zugeordnet. Beim Anlegen einer neuen Verbindung, wird die Netzadresse mit der höchsten existierenden Adresse + 1 initialisiert.
Verbindungsname	Frei wählbarer Name zur leichteren Unterscheidung der Verbindungen. **Achtung:** Der Verbindungsname darf keines der folgenden Zeichen enthalten: `{}|&~![]()"';=`
Primäre IP-Adresse	Primäre IP-Adresse der Steuerung, mit der kommuniziert wird.
Portnummer	Portnummer der Steuerung. Default für Open Modbus TCP/IP: 502
Sekundäre IP-Adresse	Sekundäre IP-Adresse der Steuerung, mit der kommuniziert wird. Schlägt eine Verbindung zur ersten Adresse fehl, wird versucht, sie über diese aufzubauen. **Hinweis:** Eine notwendige Redundanzumschaltung erfolgt bei Verbindungsausfall nach Ablauf der konfigurierten Kommunikations-Timeoutzeiten. Dadurch wird eine notwendige Redundanzumschaltung bereits vor einem erfolglosen Verbindungsaufbau erkannt.
Modbus Unit ID	`Modbus Unit ID` (Slaveadresse) - die Adresse des Empfängers
IED Typ	Auswahl des IED-Typs (IED = Intelligent Electronic Device). Auswahl aus Dropdownliste: ▸ `AREVA MiCOM P126/127` ▸ `COSTRONIC DFB` ▸ `GE Multilin F650` ▸ `GE Multilin UR-series` ▸ `ICE NPx800` ▸ `Schneider SEPAM` ▸ `None (disabled) - Standard Modbus TCP/IP` **Default**: `None (disabled)`
Neu	Legt eine neue Verbindung mit Default-Einstellungen an.
Löschen	Löscht die in der Verbindungsliste markierte Verbindung.
Bearbeiten	Öffnet die ausgewählte bestehende Verbindung zur Bearbeitung.
Speichern	Speichert eine neue oder bearbeitete Verbindung.

Abbrechen	Beendet das Bearbeiten der Verbindungseinstellungen ohne Änderungen zu speichern.

ERWEITERUNG BEI AUSWAHL COSTRONIC DFB

Parameter	Beschreibung
Basisregister Mailbox	Adresse von `ControleVie`.
Basisregister Events	Basisadresse von `tableEvenments`.
Anzahl der Register	Länge des Arrays, der mit `tableEvenments` verbunden ist (Anzahl der Eventbits/16).

NEUE VERBINDUNG ANLEGEN

1. klicken Sie auf die Schaltfläche **Neu**
2. tragen Sie die Verbindungsparameter **Netzadresse**, **Verbindungsname, IP-Adresse** und **Modbus Unit ID** ein
3. wählen Sie bei Bedarf den **IED Typ**
4. klicken Sie auf **Speichern**

Hinweis: Ist beim Anlegen bereits eine bestehende Verbindung ausgewählt, wird deren Konfiguration kopiert.

VERBINDUNGSPARAMETER EINER VERBINDUNG ANZEIGEN

1. wählen Sie in der Verbindungsliste die gewünschte Verbindung mit dem Mauszeiger aus
2. die Parameter werden angezeigt

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

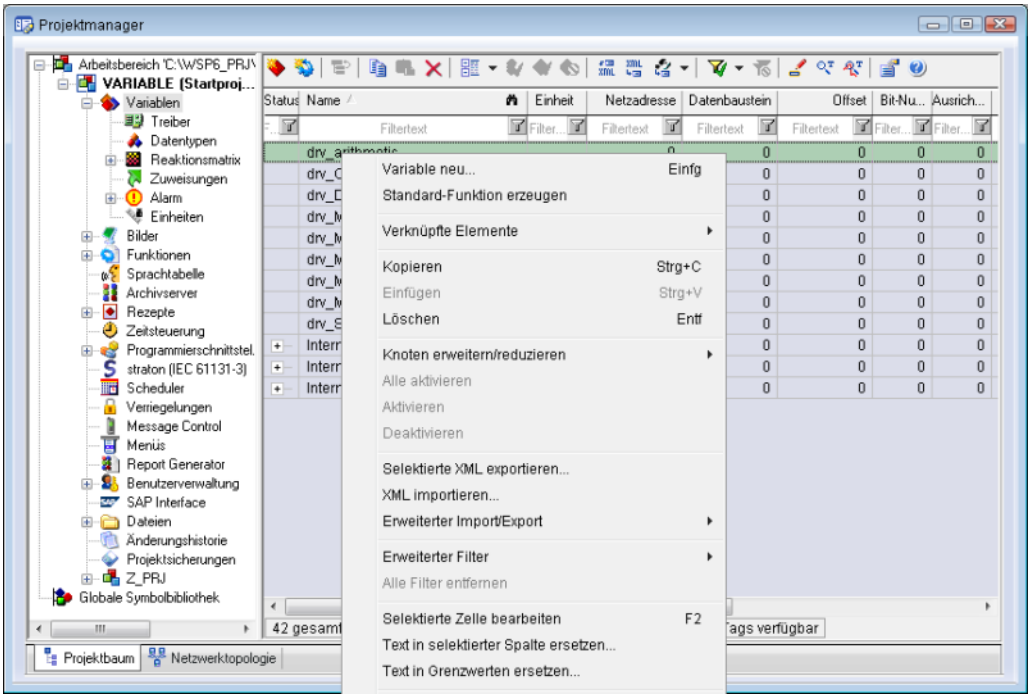
Variablen können angelegt werden:

- 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:

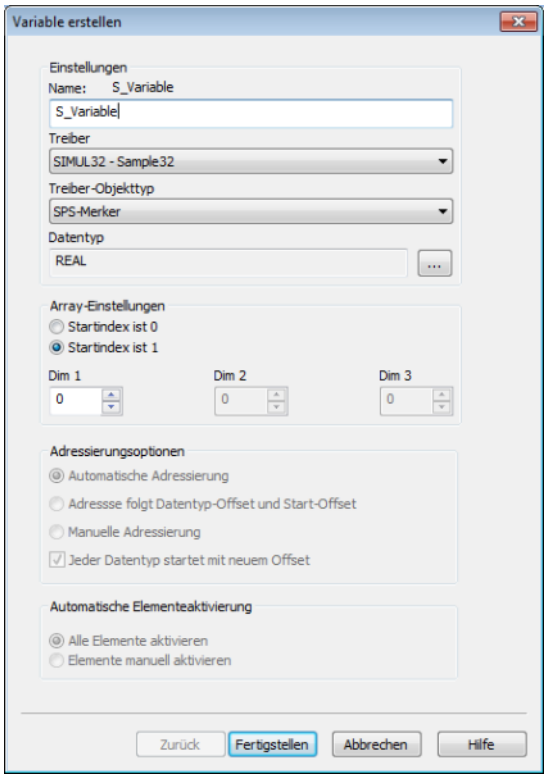
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



   Der Dialog zur Konfiguration der Variable wird geöffnet.

2. Konfigurieren Sie die Variable.

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



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.

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

Die Adressierung der Variablen definieren Sie im Eigenschaftenfenster:

Gruppe/Eigenschaft	Beschreibung
Allgemein	Gruppe mit allgemeinen Eigenschaften.
Name	Frei vergebbarer Name. Achtung: Je zenon Projekt muss der Name eindeutig sein.
Kennung	Frei vergebbare Kennung. Z. B. für **Betriebsmittelkennung**, Kommentar usw.
Adressierung	

Netzadresse	Diese Adresse bezieht sich auf die Netzadresse der Verbindungsprojektierung im Treiber. Gibt an, auf welcher Steuerung sich die Variable befindet.
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Die Bedeutung ist von der Einstellung des Treiberobjekttyps abhängig. ▸ Für Variablen des Treiberobjejkttyps **Register** bzw. **SOE - Register event**: Referenznummer des Registers [0 bis 65535] ▸ Für Variablen des Treiberobjekttyps **SOE - Numbered event**: Identifiziert den Event. Wertbereich hängt von Steuerung ab, z.B.: - AREVA MiCOM P125/P126/P127: 1 bis 127 o.162 - Costronic DFB: 0 bis (Anzahl Register)*Bit-1 Beispiel bei 63 Register: **63*16-1**=**1007**; Bereich **0** bis **1007** - GE Multilin F650: 0 bis 191 - GE Multilin UR: 0 bis Anzahl der FlexOperandStates - ICE NPx800: von 8000h - Schneider SEPAM: 1000h bis 105Fh
Ausrichtung	Ausrichtung für Variablen mit Bytelänge 1. Es kann zwischen niederwertigem und höherwertigem Byte gewählt werden.
Bitnummer	Nummer des Bits innerhalb des eingestellten Offsets. Mögliche Eingabe: 0 ... 15
Stringlänge	Nur verfügbar bei String-Variablen. Maximale Anzahl von Zeichen, die die Variable aufnehmen kann.
Treiber Anbindung/Treiberobjekttyp	Ermöglicht die Änderung des Treiberobjektyps, der bei der Erstellung der Variablen ausgewählt wurde.
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 Treiberobjekttypen stehen in diesem Treiber zur Verfügung:

Treiber-objekttyp	Kanaltyp	Lesen: Modbus Funktion (Code hex/dec)	Schreiben: Modbus Funktion (Code hex/dec)	Unterstützte Datentypen	Kommentar
Register	8	0x03/3	0x10/16	BOOL, DINT, UDINT, INT, UINT, LINT, ULINT, REAL, LREAL, SINT, USINT, STRING	Class 0 - multiple registers. Lineare Adressierung: ▸ 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**
File record	68	0x14	0x15	String	Adressierung: Der String wird über die Netzadresse (Verbindungsnummer), den Datenbaustein (Nummer des File Records) und den Offset (Nummer des Records im File Record) addressiert. **Hinweis:** Bitte beachten Sie die Informationen im Kapitel File Record (auf Seite 34).
Coil	65	0x01/1	0x05/5	BOOL	Class 1 - coils. Lineare Adressierung einstufig über **Offset**.
File transfer	12	X	X	STRING	SPS-spezifisch.
Input discrete	66	0x02/2	N/A	BOOL	Class 1 - input discretes. Lineare Adressierung einstufig über **Offset**.
Input register	64	0x04/4	N/A	BOOL, SINT, USINT, INT, UINT, DINT,	Class 1 - input registers. Lineare Adressierung wie Register.

				UDINT, REAL, LREAL, STRING	
SOE - Numbered vent	10	X	N/A	BOOL, STRING	SPS-spezifisch. Sequence of Events (Event-Puffer lesen): Adressierung der Events erfolgt nach der Event-Nummer.
SOE - Register event	11	X	N/A	BOOL, USINT, UINT	SPS-spezifisch. Sequence of Events (Event-Puffer lesen): Adressierung nach der dem Event zugeordneten Adresse des entsprechenden Modbus Registers. Nicht geeignet für COSTRONIC.
Status	67	X	N/A	UINT, INT	Status-Variablen. Details siehe Kapitel Treiberspezifische Funktionen (auf Seite 49).
Kommunikationsdetails	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variablen zur statistischen Auswertung der Kommunikation. **Hinweis:** Übermittelt zwischen Runtime und Treibern, nicht zur SPS. Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 44).

Legende:

X => wird unterstützt

-- => wird nicht unterstützt

Achtung

Bitte beachten Sie:
Werden in zenon Werte auf eine Stringvariable geschrieben, so wird bei einer projektieren **Stringlänge** von z.B. 130 Zeichen nur eine Zeichenlänge von 129 Zeichen auf die SPS weitergegeben. Das letzte Zeichen wird mit einem Null-Terminator ersetzt.

Funktionscodes

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.
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.

File Record

LESE- UND SCHREIBZUGRIFF

Der Inhalt eines „`File Record`" wird als hexadezimaler String vom Treiber beim Lesen bereitgestellt bzw. zum Schreiben empfangen:

- Enthält der „`File Record`" den String „12345" wird der Treiber diesen als hexadezimalen String „3132333435" an die Applikation weiterreichen.
- Soll der „`File Record`" den String „abcde" enthalten, muss die Applikation dem Treiber den String „6162636465" zum Schreiben übergeben.

TREIBEROBJEKTTYP

Der String wird über die Netzadresse (Verbindungsnummer), den Datenbaustein (Nummer des `File Records`) und den Offset (Nummer des `Records im File Record`) adressiert.

VARIABLE LESEN

Variable des Typs `File record` werden pollend gelesen.

Es wird immer die definiert Länge des Strings von der SPS gelesen:

- Bei einer gegebenen Stringlänge von 10 werden 5 Bytes aus der SPS gelesen, als hexadezimaler String codiert (somit ergibt sich die Länge 10) und an die Applikation weitergereicht.
- Ist die Stringlänge eine ungerade Zahl, wird zum Lesen der Daten von der SPS aufgerundet, es wird aber das niederwertige Nibble des letzten Bytes verworfen.

VARIABLE SCHREIBEN

Beim Schreiben bestimmt die Länge des an den Treiber übergebenen Strings die Anzahl der tatsächlich geschriebenen Daten.

Zu beachten ist, dass die kleinste Dateneinheit, die geschrieben werden kann, ein Wort – 2 Bytes bzw. 16 Bit – ist. Somit sollte die Stringlänge immer ein Vielfaches von 4 sein. Ist das nicht der Fall, werden überzählige Datenbytes im String ignoriert!

Ebenso ist zu beachten, dass der String hexadezimal codiert sein muss. Es sind also nur die Ziffern 0 bis 9 (Wertigkeit 0 bis 9), sowie die Buchstaben A bis F (Wertigkeit 10 bis 15) erlaubt. Die Buchstaben können klein oder groß geschrieben sein. Wird im Treiber bei der Konvertierung ein Zeichen erkannt, das nicht verwendet werden kann, erfolgt kein Schreibzugriff auf die SPS.

KONSTANTE

Konstante	Wert	Beschreibung
MB_MIN_FREC_FILENO	1	Kleinste gültige File Nummer
MB_MAX_FREC_FILENO	65535	Größte gültige File Nummer
MB_MIN_FREC_RECNO	0	Kleinste gültige Record Nummer.
MB_MAX_FREC_RECNO	65535	Größte gültige Record Nummer.
MB_FREC_MINSTRLEN	4	Minimale Stringlänge – dieser Wert darf nicht unterschritten werden, wenn Daten geschrieben werden sollen.
MB_FREC_MAXSTRLEN	476	Maximal Stringlänge.

7.3.2 Zuordnung der Datentypen

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

Steuerung	zenon	Datenart
Register / Event	BOOL	8
Register	USINT	9
Register	SINT	10
Register	UINT	2
Register	INT	1
Register	UDINT	4
Register	DINT	3
Register	ULINT	27
Register	LINT	26
Register	REAL	5
Register	LREAL	6
Register	STRING	12
-	WSTRING	21
-	DATE	18
-	TIME	17
-	DATE_AND_TIME	20
-	TOD (Time of Day)	19

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 Export-Datei in das Projekt importiert.

Tipp

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

7.4.2 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

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

IMPORT DBF-DATEI

Um den Import zu starten:

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

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

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

EXPORT DBF-DATEI

Um den Export zu starten:

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

Achtung

DBF-Dateien:

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

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

Info

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

DATEIAUFBAU DER DBASE EXPORTDATEI

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

Achtung

dBase unterstützt keine Strukturen oder Arrays (komplexe Variablen).

DBF-Dateien müssen:

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

STRUKTUR

Bezeichnung	Typ	Feldgröße	Bemerkung
KANALNAME	Char	128	Variablenname. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
KANAL_R	C	128	Ursprünglicher Name einer Variablen, der durch den Eintrag unter VARIABLENNAME ersetzt werden soll (Feld/Spalte muss manuell angelegt werden). Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
KANAL_D	Log	1	Variable wird bei Eintrag 1 gelöscht (Feld/Spalte muss manuell angelegt werden).
TAGNR	C	128	Kennung. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
EINHEIT	C	11	Technische Maßeinheit
DATENART	C	3	Datentyp (z. B. Bit, Byte, Wort, ...) entspricht dem Datentyp.
KANALTYP	C	3	Speicherbereich in der SPS (z. B. Merkerbereich, Datenbereich, ...) entspricht Treiberobjekttyp.
HWKANAL	Num	3	Netzadresse
BAUSTEIN	N	3	Datenbaustein-Adresse (nur bei Variablen aus den Datenbereich der SPS)
ADRESSE	N	5	Offset
BITADR	N	2	Für Bit-Variablen: Bitadresse Für Byte-Variablen: 0=niederwertig, 8=höherwertig Für String-Variablen: Stringlänge (max. 63 Zeichen)
ARRAYSIZE	N	16	Anzahl der Variablen im Array für Index-Variablen ACHTUNG: Nur die erste Variable steht voll zur Verfügung. Alle folgenden sind nur über VBA oder den Rezeptgruppen Manager zugänglich

LES_SCHR	L	1	Lese-Schreib-Berechtigung 0: Sollwert setzen ist nicht erlaubt 1: Sollwert setzen ist erlaubt
MIT_ZEIT	L	1	Zeitstempelung in zenon (nur wenn vom Treiber unterstützt)
OBJEKT	N	2	Treiberspezifische ID-Nummer des Primitivobjekts setzt sich zusammen aus TREIBER-OBJEKTTYP und DATENTYP
SIGMIN	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) 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. Freier String für Export und Anzeige in Listen. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
ADJWVBA	L	1	Nichtlineare Wertanpassung: 0: Nichtlineare Wertanpassung wird verwendet

			1: Nichtlineare Wertanpassung wird nicht verwendet
ADJZENON	C	128	Verknüpftes VBA-Makro zum Lesen der Variablenwerte für die nichtlineare Wertanpassung.
ADJWVBA	C	128	Verknüpftes VBA-Makro zum Schreiben der Variablenwerte für die nichtlineare Wertanpassung.
ZWREMA	N	16	Verknüpfte Zählwert-Rema.
MAXGRAD	N	16	Maximaler Gradient für die Zählwert-Rema.

Achtung

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

GRENZWERTDEFINITION

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

Bezeichnung	Typ	Feldgröße	Bemerkung
AKTIV1	L	1	Grenzwert aktiv (pro Grenzwert vorhanden)
GRENZWERT1	F	20	technischer Wert oder ID-Nummer der verknüpften Variable für einen dynamischen Grenzwert (siehe VARIABLEx) (wenn unter VARIABLEx `1` steht und hier `-1`, wird die bestehende Variablenzuordnung nicht überschrieben)
SCHWWERT1	F	16	Schwellwert für den Grenzwert
HYSTERESE1	F	14	wird nicht verwendet
BLINKEN1	L	1	Blinkattribut setzen
BTB1	L	1	Protokollierung in CEL
ALARM1	L	1	Alarm
DRUCKEN1	L	1	Druckerausgabe (bei CEL oder Alarm)
QUITTIER1	L	1	quittierpflichtig
LOESCHE1	L	1	löschpflichtig
VARIABLE1	L	1	dyn. Grenzwertverknüpfung der Grenzwert wird nicht durch einen absoluten Wert (siehe Feld GRENZWERTx) festgelegt.
FUNC1	L	1	Funktionsverknüpfung
ASK_FUNC1	L	1	Ausführung über die Alarmmeldeliste
FUNC_NR1	N	10	ID-Nummer der verknüpften Funktion (steht hier -1, so wird die bestehende Funktion beim Import nicht überschrieben)
A_GRUPPE1	N	10	Alarm/Ereignis-Gruppe
A_KLASSE1	N	10	Alarm/Ereignis-Klasse
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Farbe als Windowskodierung
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** (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
- Variablen 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 Bei Treiberstop, hat die Variable den Wert TRUE und ein **OFF**-Bit. Nach dem Treiberstart, hat die Variable den Wert FALSE und kein **OFF**-Bit.
SimulRTState	UDINT	60	Informiert über Status der Runtime bei Treibersimulation.

ConnectionStates	STRING	61	Interner Verbindungsstatus des Treibers zur SPS. Verbindungszustände: 0: Verbindung OK 1: Verbindung gestört 2: Verbindung simuliert Formatierung: <Netzadresse>:<Verbindungszustand>;…;…; Eine Verbindung ist erst nach dem ersten Anmelden einer Variablen bekannt. Damit eine Verbindung im String enthalten ist, muss einmal eine Variable dieser Verbindung angemeldet worden sein. Der Zustand einer Verbindung wird nur aktualisiert, wenn eine Variable der Verbindung angemeldet ist. Ansonsten wird nicht mit der entsprechenden Steuerung 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. 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:

BLOCKWRITE

Blockwrite ermöglicht das effizientere Absetzten von mehreren Sollwerten (z. B. Rezepte). Variablen, die im Steuerungsspeicher hintereinander liegen, werden dabei mit einem einzigen Schreibtelegramm beschrieben bzw. werden sie bei größeren Bereichen in wenige Telegramme zusammengefasst.

Achtung: bei aktiviertem Blockwrite muss die Schreibreihenfolge von Variablen nicht der Reihenfolge des Absetzens entsprechen.

Blockwrite aktivieren Sie mit einem Eintrag in der project.ini:

1. markieren Sie das Projekt im Projektmanager
2. drücken Sie die Tastenkombination `Strg`+`Alt`+`E`
3. das SQL-Verzeichnis von zenon öffnet sich im Explorer:

   `C:\ProgramData\COPA-DATA\[SQL-Ordner]\[UID]\FILES`
4. navigieren Sie zu `\zenon\system\`
   - öffnen Sie die project.ini mit einem Texteditor
   - fügen Sie folgenden Eintrag hinzu:

     [MODBUS_ENERGY]
     BLOCKWRITE=1

SEQUENCE OF EVENTS

Der Treiber enthält Funktionen zum Auslesen des Ereignis-Puffers der im Kapitel MODBUS_ENERGY (auf Seite 5) genannten Steuerungen. Die Funktionalität kann über den Treiberdialog Einstellungen (auf Seite 18) aktiviert werden. Die Art der Steuerung kann nur global für den Treiber eingestellt werden.

LESEPROZEDUR

Das MODBUS Protokoll gibt einem Slave keine Möglichkeit, ohne Abfrage des Masters etwas zu melden. Die Steuerung muss also spontane Ereignisse und die genaue Zeit ihres Auftretens speichern und ihm erst nach Abfrage des Masters übergeben.

Sequence of Events (SOE) ist ein Puffer mit Ereignissen (Wert und Zeitstempel), die in einem bestimmten Speichergebiet in den Steuerungen aufbewahrt werden. Adresse, Größe, Format und Leseprozedur aus dem Puffer sind herstellerspezifisch, da auch dazu das MODBUS Protokoll selbst keine Richtlinien gibt.

Die spontane Kommunikation unterstützt der Treiber nur für die Ereignisse, die vom Puffer ausgelesen werden. Dabei erhalten nur diese Werte in der Steuerung einen Zeitstempel. Damit können diese im Leitsystem auf allen Stellen (z. B. als Alarme, in CEL, Archive) wie spontane Ereignisse zugeordnet werden.

Der Treiber überprüft das Auftreten der neuen Ereignisse zyklisch mit der höchsten (von allen in der Treiberkonfiguration verwendeten) Priorität.

DER INITIALZUSTAND DER EREIGNISVARIABLEN

Beim Verbinden mit der Steuerung holt sich der Treiber den Initialzustand für Events aus dem Ereignispuffer (Steuerung spezifisch). Wenn ein Ereignis in der Steuerung eine bestimmte Adresse im MODBUS-Register entspricht und es wurde als `SOE - Register event` projektiert, so ist es für den Treiber möglich den Initialwert zusätzlich mit dem entsprechenden Register übereinzustimmen. Wenn aber der Initialzustand der Ereignisvariable vom Register (und nicht vom Ereignispuffer) ausgelesen wurde und der ursprüngliche Zeitstempel des Ereignisses ist nicht mehr in der Steuerung vorhanden, so wird diese Variable mit der lokalen PC-Zeit gestempelt und bekommt den Statusbit T_INTERN `(Echtzeit Intern)`.

STATUS DER TCP/IP VERBINDUNG

Mithilfe einer Variablen vom Objekttyp `Status` kann der Status der TCP/IP Verbindung zur Steuerung gelesen werden. Die Variablen haben Datentyp, **Offset** `0` und die entsprechende **Netzadresse**. Die Variable bezieht sich auf beide IP Adressen, falls die Sekundärverbindung in der Treiberkonfiguration definiert wurde. Typ: INT bzw. UINT.

Bit	Bedeutung
`Bit 1`	Zeigt an, ob die Primärverbindung aktiv ist (wird bei der Kommunikation verwendet).
`Bit 2`	Verbindungsaufbau zur primären IP-Adresse.
`Bit 3`	Verbindungsaufbau zur primären IP-Adresse fehlgeschlagen.
`Bit 5`	Zeigt an, ob die Sekundärverbindung aktiv ist (wird bei der Kommunikation verwendet).
`Bit 6`	Verbindungsaufbau zur alternativen IP-Adresse.
`Bit 7`	Verbindungsaufbau zur alternativen IP-Adresse fehlgeschlagen.

TREIBERSPEZIFISCHES KOMMANDO "SWITCHCONNECTION"

Mit dem treiberspezifischen Kommando (auf Seite 59) `SwitchConnection <HwNb>` wird die aktive IP-Adresse für eine **Netzadresse** geschlossen ohne ein INVALID-Bit zu setzen. Dann wird für die ausgewählte Verbindung eine Verbindung aufgebaut. Falls der Verbindungsaufbau fehlschlägt, wird das INVALID-Bit gesetzt und es wird zur aktiven Verbindung zurückgeschaltet.

Beispiel: `SwitchConnection 1`

AREVA MICOM P125/P126/P127

Nach dem Kommunikationsstart holt der Treiber alle bereits in der Steuerung vorhandene Ereignisse aus dem Puffer (Register Page 35H). Danach erfolgen die Abrufe (mit Quittierung) der gespeicherten Ereignisaufzeichnung von der Page 36H.

Für nicht im Puffer vorhandene Register Events wird, falls auf Page 0H vorhanden, der Initialwert aus entsprechendem MODBUS-Register ausgelesen und mit lokaler PC-Zeit gestempelt.

Nicht im Puffer vorhandene `Numbered Events` bekommen den Initialwert 0 und werden mit lokaler PC-Zeit gestempelt.

GE MULTILIN F650

Nach dem Kommunikationsstart des Treibers holt er den Initialzustand der Ereignissen aus dem Register (Adresse 0xF000). Danach erfolgen die Abrufe aus dem Ringpuffer (0xFD00).

Die von der Steuerung gemeldeten Ereignisse kommen schon mit dem Zeitstempel der Steuerung und bekommen auch das Statusbit T_EXTERN `(Echtzeit Extern)`.

SCHNEIDER SEPAM

Events werden im Bereich „SOE Numbered Events" angelegt. Als Offset wird die Coil-Bit Adresse der Einstellung angegeben, die spontan gelesen werden soll. Unterstützt werden die Adressen 1000h – 105Fh. Das Auslesen des Ausgangszustands nach Neuverbindung wird unterstützt.

DER TREIBER OBJEKTTYP DES EREIGNISSES

Für die Ereignisse kommen zwei mögliche **Treiber Objekttypen** in Frage: `SOE - Numbered event` und `SOE - Register event`, wobei nicht beide für jede Steuerung eine Funktion besitzen.

AREVA MICOM P125/P126/P127

Für Areva MiCOM sind beide Objekttypen - `Numbered event` und `Register event`, unterstützt.

Das `SOE - Numbered event` entspricht einem Event das nur bei Auftreten einer Bedingung generiert wird (und bei Verschwinden nicht). Die Identifikation eines Events wird über die **Offset** Einstellung in der Variablenadressierung festgelegt (z. B. Offset 5 für Event 05). Tritt ein Event in der Steuerung auf, wird die assoziierte BOOL Variable auf 1 gesetzt und vom Treiber sofort wieder auf 0. Der kommende Wert kann durch Anlegen von Alarm/CEL/Archive oder einer Funktion für Grenzwertverletzung ausgewertet werden.

Die `Register events` entsprechen dem Rest der Events. Die `Register events` sind in der AREVA Steuerung einem bzw. mehreren Bits in einem bestimmten Register der Page 0 zugeordnet (z. B. Offset 20h, Bit 0). Das heißt die Steuerung sorgt dafür dass das Event die Änderung eines Zustandes in einem Register spiegelt. In der Variablenadressierung erfolgt die Identifikation des Ereignisses über **Offset** und

Bit und beschreibt die assoziierten Datenpunkte in einem Page 0 Register. Als Datentyp kann BOOL, USINT und UINT gewählt werden.

Welche Ereignisse von der AREVA Steuerung unterstützt werden und den Ereignistyp entnehmen sie der Dokumentation der AREVA Steuerung.

GE MULTILIN F650

Für GE Multilin F650 ist nur der Objekttyp `SOE - Numbered event` unterstützt.

Über den Treiber Objekttyp `SOE - Numbered event` kann jedem Event eine BOOL Variable zugeordnet werden. Die Identifikation des Events wird über die **Offset** Einstellung der Variable festgelegt (0 bis 191).

Der Treiber Objekttyp `SOE - Register event` hat für die GE Multilin F650 Steuerung keine Funktion.

8.1 ICE NPx800

Das Lesen der Events wird nur für den vereinfachten Event-Modus der NPx800 unterstützt: nur Ein bzw. Wischer-Events, keine Status-Events (Ein/Aus).

Event-Variablen werden unter dem Treiberobjekttyp `SOE-Numbered Events` als BOOL angelegt. Die Event-Parameter können durch das Anlegen einer Stringvariable mit gleichen Treiberobjektyp und Offset gelesen werden.

Achtung: Die Stringvariable hat nur dann einen gültigen Wert, wenn der Event gerade ausgelöst wird, ansonsten enthält sie einen leeren String. Sie verhält sich daher gleich wie auch die BOOL Variablen, d.h. wie ein Wischer-Event. Der zenon Alarm/CEL-Eintrag kann also über die BOOL oder auch über die String Variable ausgelöst werden. Das Anzeigen des Strings ist jedoch nur über einen dynamischen Grenzwerttext (Variablen.chm::/15311.htm) möglich.

Die Eventnummer wird über das Offset konfiguriert.

WERT DER EVENT-VARIABLEN

Der Ablauf beim Auftreten eines Events:

1. Die Parameter werden in eine evtl. vorhandene Stringvariable geschrieben
2. Die evtl. vorhandene Bool Variable wird 1 gesetzt
3. Die evtl. vorhandene Bool Variable wird 0 gesetzt
4. Die evtl. vorhandene Stringvariable mit den Parametern wird auf die Länge 0 gesetzt

PARAMETERSTRING

In Stringvariablen besteht der Parameterstring aus Integerwerten die durch Strichpunkte (**;**) getrennt sind. Diese Integerwerte entsprechen folgenden Daten des Events:

- 1: Event data at word offset 2
- 2: Event data at word offset 7 (parameter 0)
- 3: Event data at word offset 8 (parameter 1)
- 4: Event data at word offset 9 (parameter 2)
- 5: Event data at word offset 10 (parameter 3)
- 6: Event data at word offset 11 (parameter 4)
- 7: Event data at word offset 12 (parameter 5)
- 8: Event data at word offset 13 (parameter 6)
- 9: Event data at word offset 14

ABLAUF EVENTS

Ablauf bei einem Event:

- Beim Starten der Runtime oder beim Wiederherstellen der Treiberverbindung werden alle Events mit `0` initialisiert.
- Wird ein Event empfangen, dann wird:
  - eine Stringvariable mit den **Event-Parametern** gefüllt (sofern angelegt)
  - der Status des **A/D** Bits der **BOOL** Variable zugewiesen

    Wird nach dem Starten der Runtime oder beim Wiederherstellen der Treiberverbindung ein `0`–Event empfangen, kann diese Wertänderung nicht wahrgenommen werden.
    Wird zweimal hintereinander ein `1`-Event gesendet (ohne `0`-Event), kann der zweite Event nicht wahrgenommen werden.
  - die Stringvariable zurückgesetzt (Leerstring)

Die Stringvariable kann auch mit den Event-Parametern ausgewertet werden. Im ersten Parameter ist auch der Wert des **A/D** Bits vorhanden. Über die Stringvariable können alle Events in der RT wahrgenommen werden.

ÜBERPRÜFUNG EV-FLAG

Der Treiber überprüft für den ICELEC Alarm Stack das EV-Flag. Wird der Wert `False` zurückgegeben, werden die gelesenen Datzen ignoriert und keine weiteren Events ausgelesen.

DATEITRANSFER DISTURBANCE RECORDS

Für den Dateitransfer von **Disturbance Records** kann eine String-Variable angelegt werden. Mit dieser ist das Lesen und Löschen von Dateien möglich.

Eine Datei wird dabei über die Nummer des **Disturbance record descriptor** identifiziert.
Mögliche Werte: `1`, `2`, `3`, `4`

Die Records sind im **COMTRADE**-Format und haben eine laufende Nummer (`0` - `255`).. Die Beschreibung des Records wird in den Deskriptoren (`1` - `4`) abgelegt. Diese enthalten Datum, Uhrzeit laufende Nummer, Quelle und Position. Es können maximal 4 Disturbance Records gespeichert werden.

Der Zielordner wird bei der Treiberkonfiguration in der Registerkarte Einstellungen (auf Seite 18) festgelegt.

HOCHLADEN UND LÖSCHEN

- Datei hochladen: Eintrag **GET x** in der String-Variablen lädt die entsprechende Datei hoch.
- Datei löschen: Eintrag **DEL x** in der String-Variablen löscht die entsprechende Datei.

x steht für die **Disturbance record descriptor** Nummer (`1` - `4`)

DATEINAME

Die Datei wird nach folgendem Schema benannt

A_yyyy-mm-dd_hh-mm-ss-MMM_DescriptorZ_(x-y)

Bedeutung:

- **A**: Hardware-Adresse des Geräts
- **yyyy:** Jahr der Erstellung
- **mm:** Monat der Erstellung
- **hh:** Stunde der Erstellung
- **mm:** Minute der Erstellung
- **ss:** Sekunde der Erstellung
- **MMM**: Millisekunde der Erstellung
- **Z**: laufende Nummer des Records
- **x**: Quelle
- **y**: Position

Zum Beispiel:

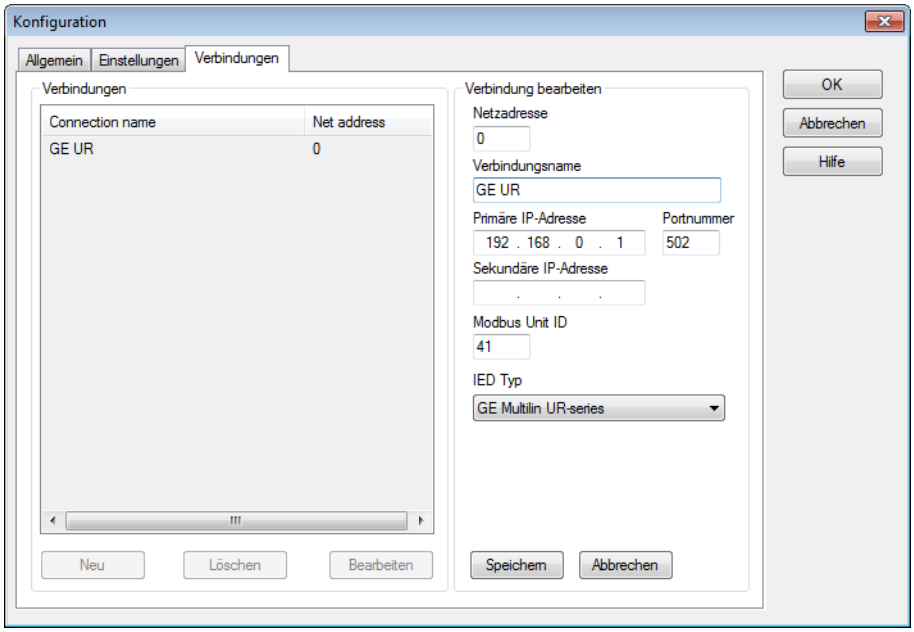
1_2009-09-07_14-01-36-690_Disturbance_109_(19-420).HDR

1_2009-09-07_14-01-36-690_Disturbance_109_(19-420).CFG

1_2009-09-07_14-01-36-690_Disturbance_109_(19-420).DAT

8.2 GE Multilin UR-series

Der IED-Typ wird in der Registerkarte Verbindungen (auf Seite 21) der Treiberkonfiguration ausgewählt:



Klick auf Dropdownliste im Bereich **IED Typ** öffnet Auswahl der IED-Typs, wählen Sie dort **GE Multilin UR-series**.

SOE UNTERSTÜTZUNG

Über die SOE-Funktionalität des Treibers können Events vom Gerät gelesen werden. Dafür wird die Datei **EVT.TXT** am Gerät ausgewertet. Events müssen vom Treiberobjekttyp `SOE - Numbered event` und vom Datentyp `BOOL` angelegt werden. Die Adressierung erfolgt über die Offset-Einstellung.

Eine Liste der Eventnummern und deren Bedeutung kann über den Web Server der UR-Serie abgerufen werden:
http://IP_Address_ of_UR_unit/FlexOperandStates.htm (z. B.:
http://192.168.37.67/FlexOperandStates.htm).

Das `SOE - Numbered event` entspricht einem Event das nur bei Auftreten einer Bedingung generiert wird (und bei Verschwinden nicht). Die Identifikation eines Events wird über die **Offset** Einstellung in der Variablenadressierung festgelegt (z. B. Offset 5 für Event cause 5). Tritt ein Event in der Steuerung auf, wird die assoziierte BOOL Variable auf 1 gesetzt und vom Treiber sofort wieder auf 0. Der kommende

Wert kann durch Anlegen von Alarm/CEL/Archive oder einer Funktion für Grenzwertverletzung ausgewertet werden.

DATEITRANSFER

Über eine Kommandovariable des Treiberobjekttyps **File transfer** (siehe Dateitransfer Beschreibung für ICE-Geräte)die Datei auf den lokalen Rechner übertragen werden. Das Kommando besteht aus: **GET** gefolgt von einem Leerzeichen und dem Dateinamen: **GET EVT.TXT**

Die Datei wird in das Dateitransferverzeichnis (wie in der Registerkarte Einstellungen (auf Seite 18) der Konfiguration definiert) kopiert.
Format des Dateinamens: `Netzadresse+Unterstrich+Dateinamen.`
Die Netzadresse wird von Treiber/Variable übernommen und nur zum Speichern verwendet, damit die Datei bei mehreren Steuerungen nicht überschrieben wird.

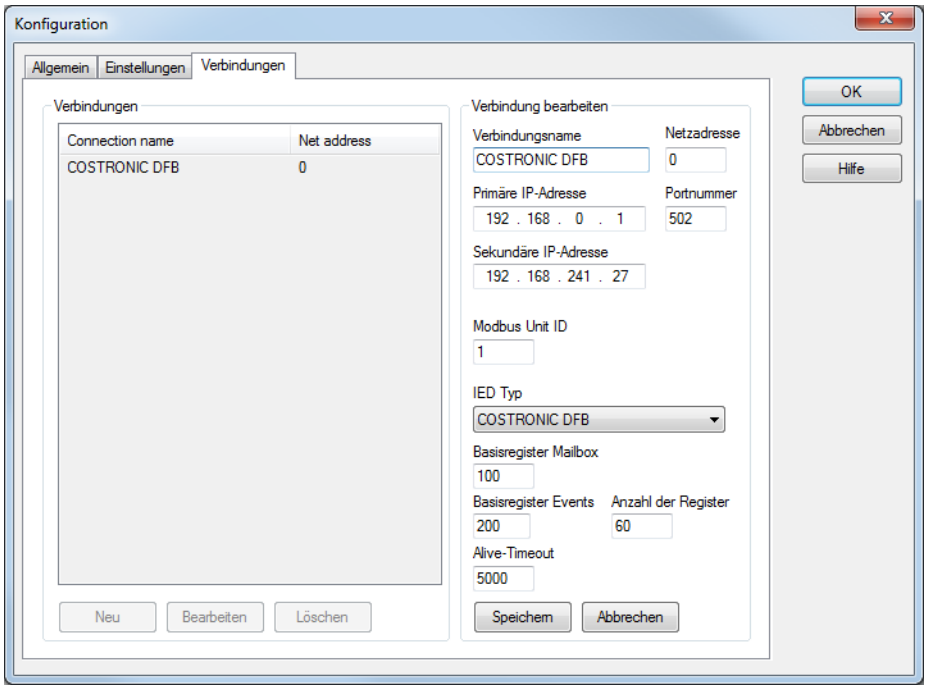
Beispiel: **0_EVT.TXT**

8.3 COSTRONIC DFB

ALIVE TIMEOUT: ERKENNUNG EINES SPS-STOPPS

Für COSTRONIC DFB ist die Erkennung eines Stopps der SPS möglich. Dazu wird ein Zähler für die Eventkommunikation in der Costronics Kommunikation-Mailbox überwacht. Ändert sich dieser Zähler während einer konfigurierbaren Wartezeit nicht, wird eine Status-Variable für die Verbindung auf `1` gesetzt. Bei der nächsten Änderung des Zählers wird sie wieder auf `0` gesetzt.

Die Konfiguration erfolgt über die Option **Alive timeout** in der Treiberkonfiguration.



Die Status-Variable muss vom Treiber Objekttyp Status sein und den **Offset** `1` haben. Sie kann folgende Werte annehmen:

Wert	Bedeutung
0	Event-Mailbox wurde noch nie gelesen.
1	`Alive` Zähler wird hochgezählt.
2	`Alive` Zähler Wartezeit abgelaufen.

Achtung: Der `Alive` Zähler ist Teil der Event-Mailbox. Wird die Event-Mailbox nicht ausgelesen (keine Variablen aktiv), wird auch der Status nicht erneuert. Das bedeutet: Damit der `Alive` -Status überwacht wird, muss mindestens eine Event-Variable aktiv sein.

TREIBERSPZIFISCHES KOMMANDO FÜR UPDATE EVENT-STATUS

Mit einem treiberspezifischen Kommando (auf Seite 59) kann das Einlesen der aktuellen Werte von ereignisgesteuerten Variablen erzwungen werden.

- Kommando: `UpdateEventState <HwNb>`

  Zwingt den Treiber, alle Event-Status von der SPS zu lesen.

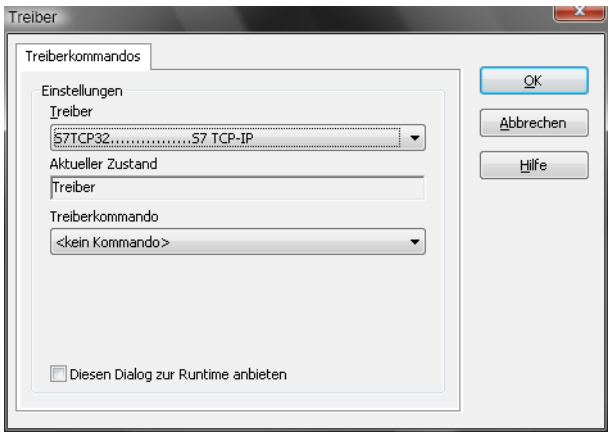
Beispiel: `UpdateEventState 1`

9. Treiberkommandos

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.

Treiberkommandos dienen dazu, Treiber über zenon zu beeinflussen, z. B. starten und stoppen.
Die Projektierung erfolgt über die Funktion **Treiber Kommandos**. Dazu:

- legen Sie eine neue Funktion an
- wählen Sie *Variablen -> Treiberkommandos*
- der Dialog zur Konfiguration wird geöffnet



Parameter	Beschreibung
Treiber	Dropdownliste mit allen im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, in aktuellen Versionen ohne Funktion.
Treiberkommando	Dropdownliste zur Auswahl des Kommandos.
▸ `Treiber starten (Online-Modus)`	Treiber wird neu initialisiert und gestartet.
▸ `Treiber stoppen (Offline-Modus)`	Treiber wird angehalten, es werden keine neuen Daten 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 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).
Diesen Dialog zur Runtime anbieten	Dialog wird zur Runtime für Änderungen angeboten.

TREIBERKOMMANDOS IM NETZWERK

Wenn sich der Rechner, auf dem die Funktion **Treiberkommandos** ausgeführt wird, im zenon Netzwerk befindet, werden zusätzliche 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 7.60 -> 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. ▸ 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, 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 60)	▸ Siehe Kapitel: Fehlernummern ▸ 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 60)	▸ Sind die Variablen korrekt adressiert? ▸ Sind die korrekten Datentypen verwendet? ▸ Ist die Wertberechnung korrekt?