zenon Treiber Handbuch

MODBUS_ENERGY

v.7.00

Inhalt

1. Willkommen bei der COPA-DATA Hilfe

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 Fanuc; Modbus RTU; Mondial; Schiele; Telemecanique; Schneider; Wago; 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	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 5.0, CE 6.0; Windows XP, Vista, 7, Server 2003, Server 2008/R2;
CE Plattformen	x86; ARM; Pocket-PC;

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

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, zB. %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

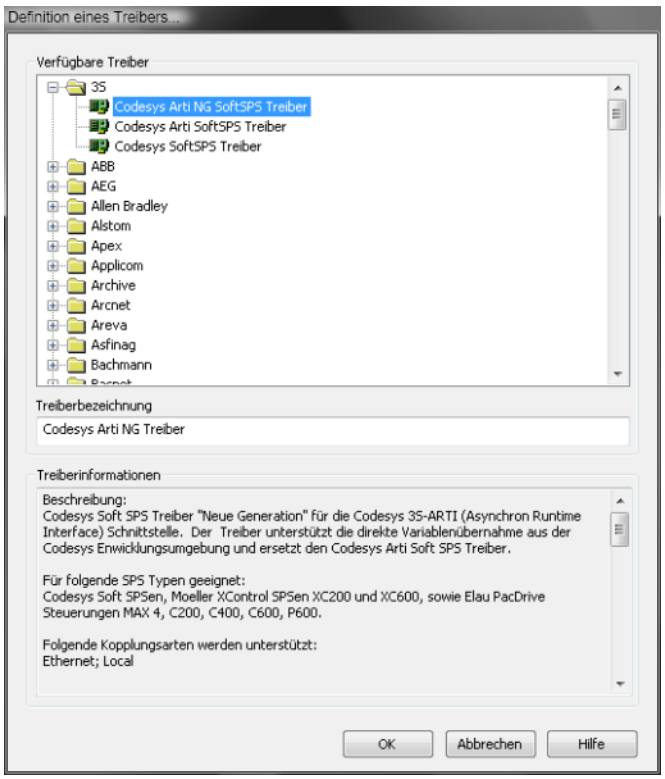
In diesem Kapitel lesen Sie, wie Sie den Treiber im Projekt anlegen und welche Einstellungen beim Treiber möglich sind.

> **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

Um einen neuen Treiber anzulegen:

- Klicken Sie mit der rechten Maustaste im Projektmanager auf **`Treiber`** und selektieren Sie im Kontextmenü **`Treiber neu`**.
- In der folgenden Dialogbox bietet Ihnen das Programm eine Auflistung aller verfügbaren Treiber an.



- Selektieren Sie den gewünschten Treiber und vergeben Sie eine Bezeichnung für diesen:
  - Die Treiberbezeichnung muss eindeutig sein, d.h. wird ein und derselbe Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
  - 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.

▶ Bestätigen Sie den Dialog mit **ok**. Im folgenden Dialog werden die einzelnen Konfigurationen der jeweiligen Treiber eingestellt.

▶ Für ein Projekt müssen nur die jeweils notwendigen Treiber eingebunden werden. Späteres Einbinden eines weiteren Treibers ist problemlos möglich.

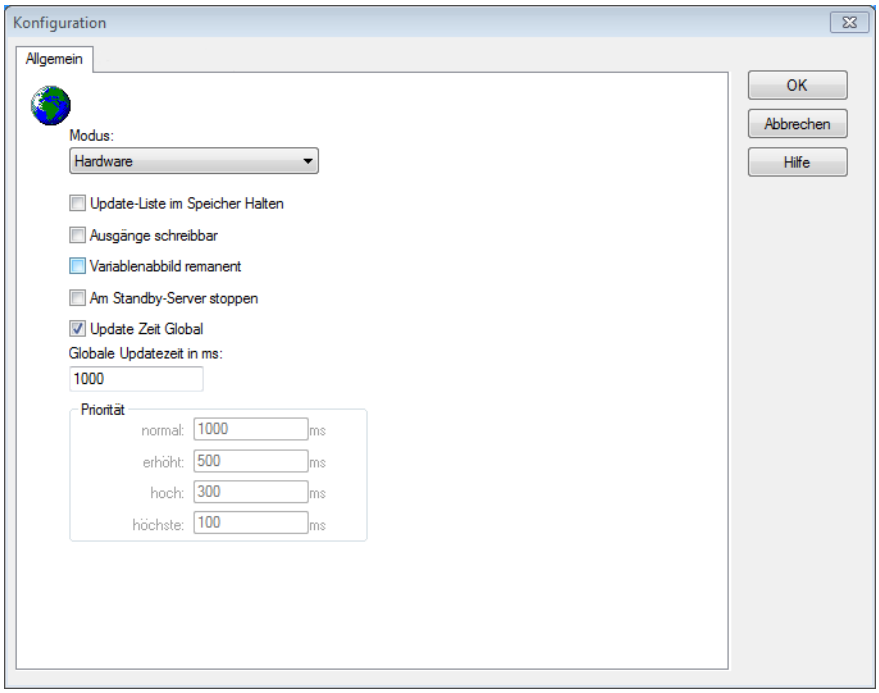
> **Info**
>
> *Bei neu angelegten Projekten oder bei bestehenden Projekten, die in eine Version ab 6.21 konvertiert werden, werden die folgenden Treiber automatisch angelegt:*
>
> - Intern
> - MathDr32
> - SysDrv.

▸

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein



Parameter	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 bzw. die Variablen behalten die über zenon Logic gesetzen Werte. Jede Variable hat seinen eigenen Speicherbereich. z.B. zwei Variablen vom Typ Merker mit Offset 79, können zur Laufzeit 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 `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 Stop-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.
`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 Prioritäten eingestellt. Alle Variablen mit der entsprechenden Priorität werden in der eingestellten Zeit gepollt. Die Zuordnung zu den Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften. Mit den Prioritäten kann die Kommunikation der einzelnen Variablen auf die Wichtigkeit bzw. benötigte Aktualität abgestuft werden. Daraus ergibt sich eine verbesserte Verteilung der Kommunikationslast.

UPDATE ZEIT ZYKLISCHE TREIBER

Für zyklische Treiber gilt:

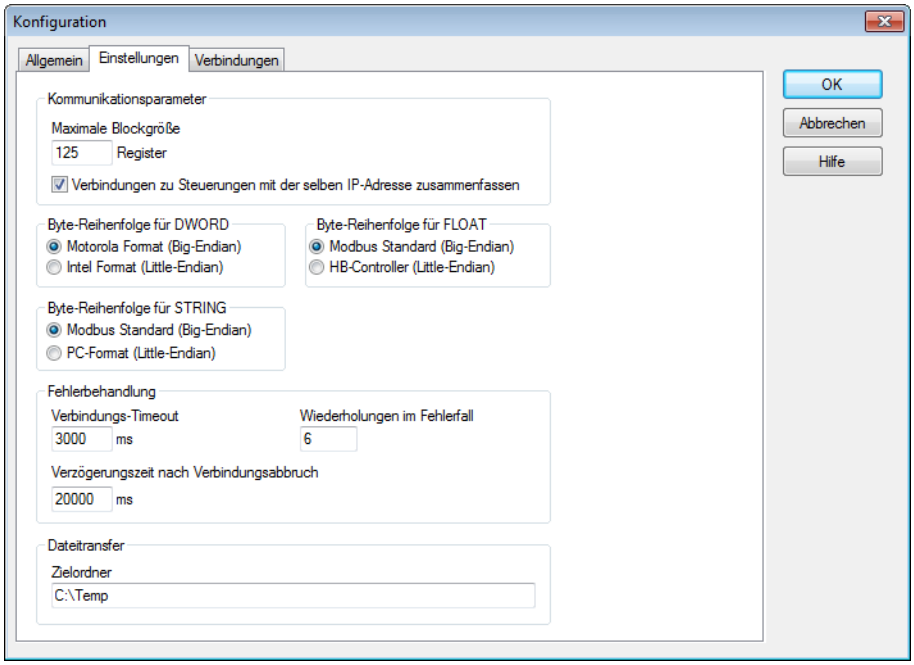
Beim **`Sollwert Setzen`**, **`Advicen`** von Variablen und bei **`Requests`** wird sofort ein Lesezyklus für alle Treiber ausgelöst - unabhängig von der eingestellten Update Zeit. Damit wird sicher gestellt, dass der Wert nach dem Schreiben in der Visualisierung sofort zur Verfügung steht. Update-Zeiten 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



Parameter	Beschreibung
Kommunikationsparameter	
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.
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	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	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	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)	Darstellung nach Modbus Standard mit vertauschten Zeichen.
PC-Format (Little-Endian)	Darstellung nach Modbus Standard mit vertauschten Zeichen.
Fehlerbehandlung	
Verbindungs-Timeout	Zeit in Millisekunden, die auf die Antwort eines Slaves gewartet wird. Kommt keine Antwort innerhalb dieser Zeit, wird ein Kommunikationsfehler ausgegeben.
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.
Zielordner	Ordner für Dateitransfer.
OK	Übernimmt Änderungen und schließt den Dialog.
Abbrechen	Verwirft Änderungen und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

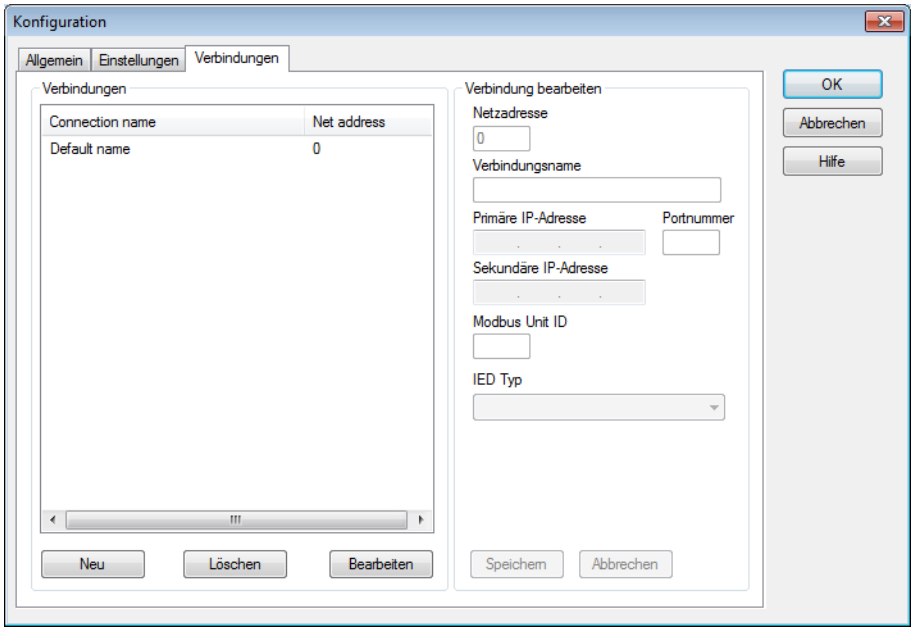
Info

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.
Modbus Unit ID	Modbus Unit ID (Slaveadresse)
IED Typ	Auswahl des IED Typs (IED = Intelligent Electronic Device). Klick öffnet Dropdownliste mit Auswahl: ▸ 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

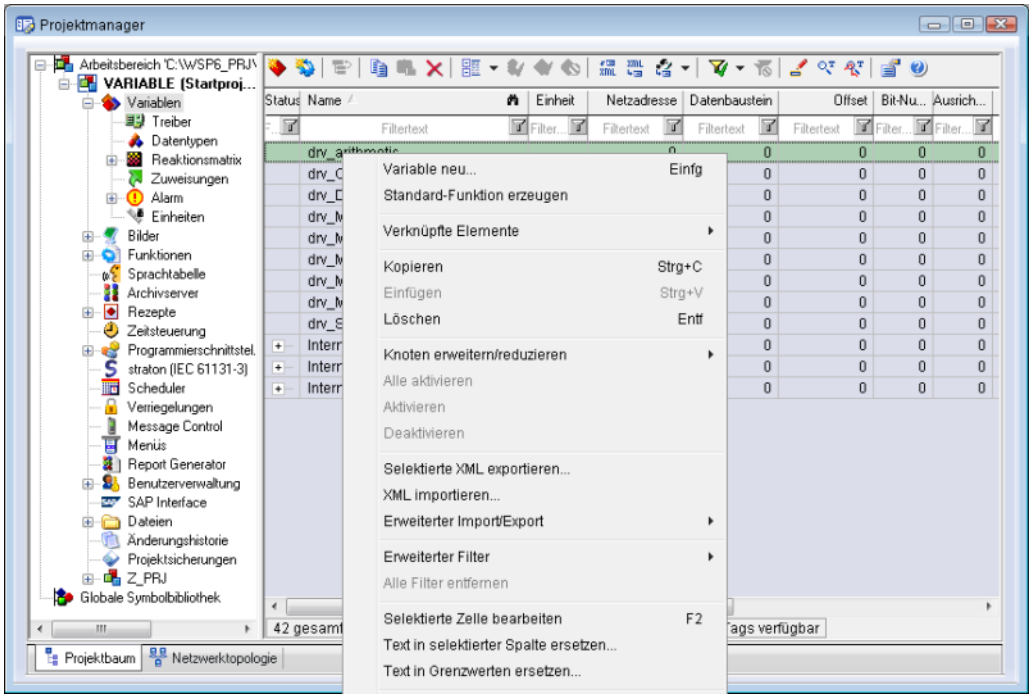
Variablen können angelegt werden:

- 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:

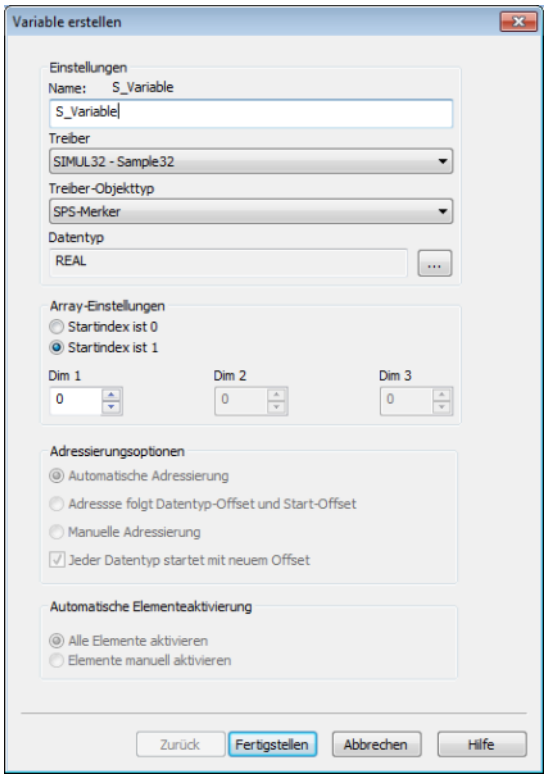
1. wählen Sie im Knoten **`Variablen`** im Kontextmenü den Befehl **`Variable neu`**



2. der Dialog zur Konfiguration der Variable wird geöffnet

3. konfigurieren Sie die Variable

4. 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. **Achtung:** Das Zeichen **#** ist im Variablennamen nicht zugelassen. Bei Verwendung nicht zugelassener Zeichen kann die Variablenerstellung nicht abgeschlossen werden, die Schaltfläche **Fertigstellen** bleibt inaktiv.
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.
Treiber-Objekttyp (cti.chm::/28685.htm)	Wählen Sie aus der Dropdownliste den passenden Treiber-Objekttyp 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.

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	
Name	Frei vergebbarer Name. **Achtung:** Je zenon Projekt muss der Name eindeutig sein.
Kennung	Frei vergebbare Kennung; für Betriebsmittelkennung, Kommentar ...
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 für AREVA MiCOM P125/P126/P127: 1 bis 127 o.162 Wertbereich für Costronic DFB: 0 bis Anzahl der Register Wertbereich für GE Multilin F650: 0 bis 191 Wertbereich für GE Multilin UR: 0 bis Anzahl der FlexOperandStates Wertbereich für Schneider SEPAM: 1000h bis 105Fh
Ausrichtung	Ausrichtung für Variablen mit Bytelänge 1. Es kann zwischen niederwertigem und höherwertigem Byte gewählt werden.
Bit-Nummer	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/Treiber-Objekttyp	Ermöglicht die Änderung des Treiber-Objekttyps, der bei der Erstellung der Variablen ausgewählt wurde.
Treiber Anbindung/Datentyp	Ermöglicht die Änderung des Datentyps, der bei der Erstellung der Variablen ausgewählt wurde. ACHTUNG: Wenn der Datentyp nachträglich geändert wird, müssen alle anderen Eigenschaften der Variablen überprüft bzw. angepasst werden.

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
Register	8	0x03/3	0x10/16	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, 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
Coil	65	0x01/1	0x05/5	BOOL	Class 1 - coils. Lineare Adressierung einstufig über Offset.
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, UDINT, REAL, LREAL, STRING	Class 1 - input registers. Lineare Adressierung wie Register.
SOE – Numbered event	10	J	N/A	BOOL	SPS-spezifisch. Sequence of Events (Event-Puffer lesen): Adressierung der Events

					erfolgt nach der Event-Nummer.
SOE - Register event	11	J	N/A	BOOL, USINT, UINT	SPS-spezifisch. Sequence of Events (Event-Puffer lesen): Adressierung nach der dem Event zugeordneten Adresse des entsprechenden Modbus Registers.
File transfer	12	J	J	STRING	SPS-spezifisch.
Treibervariable	35	J	J	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 Infos finden Sie bei den Treibervariablen (auf Seite 38).

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.
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.

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
-	ULINT	27
-	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.

7.4.1 XML Import

Für den Import/Export von Variablen gilt:

- Der Import/Export darf nicht aus dem Globalprojekt gestartet werden.
- Der Start erfolgt über:
  - Kontextmenü zu Variablen bzw. Datentyp im Projektbaum
  - oder Kontextmenü einer Variablen bzw. eines Datentyps
  - oder Symbol in der Symbolleiste Variablen

> **Achtung**
>
> *Beim Import/Überschreiben von existierenden Datentypen werden alle Variablen geändert, die auf diesem existierenden Datentyp basieren.*
>
> ***Beispiel:***
>
> *Es existiert ein Datentyp XYZ abgeleitet vom Typ `INT` mit Variablen, die auf diesem Datentyp basieren. Ihre zu importierende XML-Datei enthält ebenfalls einen Datentyp mit Namen XYZ, allerdings abgeleitet vom Typ `STRING`. Wird dieser Datentyp importiert, so wird der existierende Datentyp überschrieben und bei allen auf ihm basierenden Variablen der Typ angepasst. D.h. die Variablen sind jetzt `STRING`- und keine `INT`-Variablen mehr.*

7.4.2 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

IMPORT DBF-DATEI

Um den Import zu starten:

1. führen Sie einen Rechtsklick auf die Variablenliste aus
2. wählen Sie im Dropdownmenü 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 Dropdownmenü 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 KANALNAME 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	Datenart (z.B. Bit, Byte, Wort, ...) entspricht dem Datentyp.
KANALTYP	C	3	Speicherbereich in der SPS (z.B. Merkerbereich, Datenbereich, ...) entspricht Treiber-Objekttyp.
HWKANAL	Num	3	Bus-Adresse
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 KANALTYP und DATENART
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, bzw. 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 Alarmverwaltung
FUNC_NR1	N	10	ID-Nummer der verknüpften Funktions (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 Treibervariablen

Das Treiberkit implementiert eine Reihe von Treibervariablen. Diese sind unterteilt in:

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

Die Definitionen der im Treiberkit implementierten Variablen sind in der Importdatei **`drvvar.dbf`** (auf der CD im Verzeichnis: `CD_Laufwerk:/Predefined/Variables`) verfügbar und können von dort importiert werden.

Hinweis: Variablennamen müssen in zenon einzigartig sein. Soll nach einem Import der Treibervariablen aus **`drvvar.dbf`** ein erneuter Import durchgeführt werden, müssen die zuvor importierten Variablen umbenannt werden.

Info

Nicht jeder Treiber unterstützt alle Treibervariablen.

Zum Beispiel werden:

- Variablen für Modem-Informationen nur von modemfähigen Treibern unterstützt
- Treibervariablen für den Polling-Zyklus nur für rein pollenden Treibern
- verbindungsbezogene Informationen wie ErrorMSG nur von Treibern, die zu einem Zeitpunkt nur eine Verbindung bearbeiten

INFORMATION

Name aus Import	Typ	Offset	Erklärung
MainVersion	UINT	0	Haupt-Versionsnummer des Treibers.
SubVersion	UINT	1	Sub-Versionsnummer des Treibers.
BuildVersion	UINT	29	Build-Versionsnummer des Treibers.
RTMajor	UINT	49	zenon Hauptversionsnummer
RTMinor	UINT	50	zenon Sub-Versionsnummer
RTSp	UINT	51	zenon Servicepack-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.

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	SINT	14	Bitanzahl pro Zeichen der seriellen Schnittstelle. Wert = 0, wenn der Treiber keine serielle Kommunikation herstellen kann.
StopBit	SINT	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	UDINT	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	UDINT	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	SINT	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
4. `C:\ProgramData\COPA-DATA\[SQL-Ordner]\[UID]\FILES`
5. 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 15) aktiviert werden. Die Art der Steuerung kann nur global für den Treiber eingestellt werden.

DIE LESENSPROZEDUR

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

Sequence of Events (SOE) ist ein Puffer mit Ereignissen (Wert und Zeitstempel) die in einem bestimmten Speichergebiet in den Steuerungen aufbewahrt werden. Die Adresse, Größe, Format und

Lesensprozedur aus dem Puffer ist Herstellerspezifisch, da auch dazu das MODBUS Protokoll selbst keine Rechtlinien gibt.

Die spontane Kommunikation unterstützt der Treiber nur für die Ereignisse die vom Puffer ausgelesen werden, in dem Sinn dass nur diese Werte in der Steuerung einen Zeitstempel erhalten und somit 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)`.

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 IEC 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 Primitivtyp **`Numbered Events`** als BOOL angelegt. Die Event-Parameter können durch das Anlegen einer Stringvariable unter dem Primitivtyp **`Numbered Events`** gelesen werden.

Achtung: Die Stringvariable hat nur 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 nur über dynamischen Grenzwerttext möglich.

Die Eventnummer wird über das Offset konfiguriert.

SOLLWERT SETZEN

Der Ablauf des Sollwert-Setzens 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

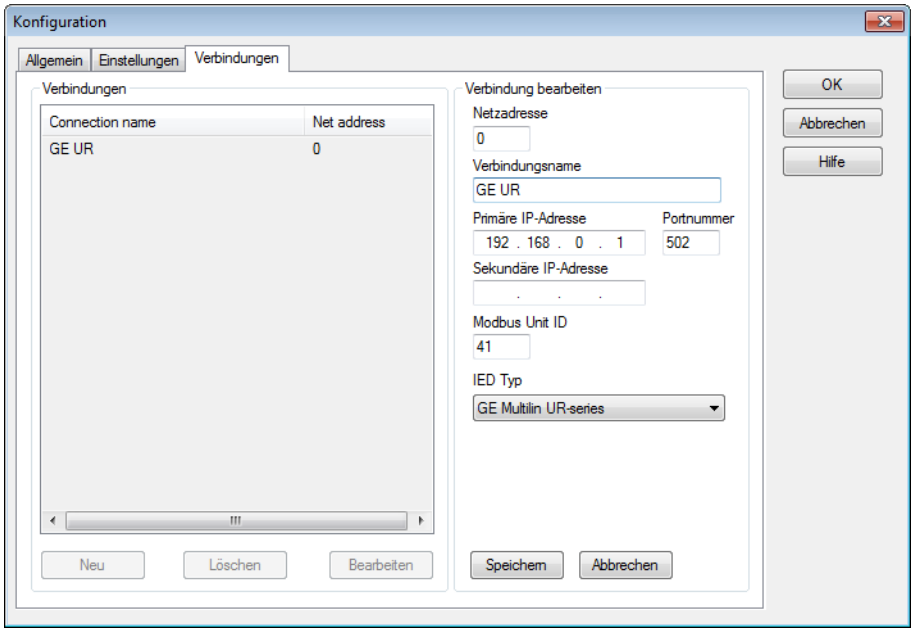
Der Aufbau des Parameterstrings (Integerwerte durch Strichpunkte getrennt):

- WORD 0: Event data at word offset 2
- WORD 1: Event data at word offset 7 (parameter 0)
- WORD 2: Event data at word offset 8 (parameter 1)
- WORD 3: Event data at word offset 9 (parameter 2)
- WORD 4: Event data at word offset 10 (parameter 3)

- WORD 5: Event data at word offset 11 (parameter 4)
- WORD 6: Event data at word offset 12 (parameter 5)
- WORD 7: Event data at word offset 13 (parameter 6)
- WORD 8: Event data at word offset 14

8.2 GE Multilin UR-series

Der IED-Typ wird in der Registerkarte Verbindungen (auf Seite 19) 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 Treiber-Objekttyp `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 Webserver 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 15) 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`**

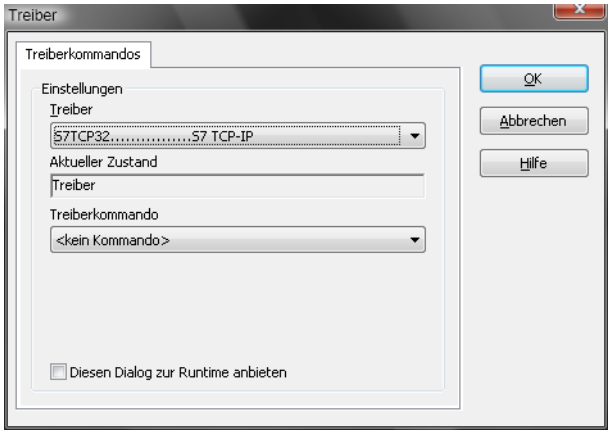
9. Treiberkommandos

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 Hilfen, 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 Diagnose Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter *Start/Alle Programme/zenon/Tools 7.00 -> Diagviewer.*

zenon Treiber protokollieren alle Fehler in Log-Dateien. Der Standardordner für die Log-Dateien ist der Ordner **LOG** unterhalb des Ordners `ProgramData`, zum Beispiel: `C:\ProgramData\zenon\zenon700\LOG` für die zenon Version 7.00 SP0. Log-Dateien sind Textdateien mit einer speziellen Struktur.

Achtung: Mit den Standardeinstellungen zeichnet ein Treiber nur Fehlerinformationen auf. Mit dem **Diagnose 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 Diagnose Viewer kann man auch:

- eben erstellte Einträge live mitverfolgen
- die Aufzeichnungseinstellungen anpassen
- den Ordner, in dem die Log-Dateien gespeichert werden, ändern

Hinweise:

1. Unter Windows CE werden aus Ressourcegründen auch Fehler standardmäßig nicht protokolliert.
2. Der Diagnose Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.
3. Der Diagnose 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.
4. 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**.
5. 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.
6. 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 **Diagnose Viewers** weiter aufgezeichnet.

Info

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

10.2 Checkliste

Überprüfen Sie bei Kommunikationsfehlern:

- Ist die Steuerung an die Stromversorgung angeschlossen?
- Sind die Teilnehmer im `TCP/IP`-Netz verfügbar?
- Kann die Steuerung über den `Ping` Befehl erreicht werden?
- Kann die Steuerung auf dem entsprechenden Port über `TELNET` erreicht werden?
- Sind Steuerung und PC mit dem passenden Kabel verbunden?
- Wurde die Netzadresse sowohl im Treiberdialog als auch in den Adresseigenschaften der Variablen korrekt eingestellt.?
- Wird in der Variable der richtige Objekttyp verwendet?
- Stimmt die Offset-Adressierung der Variablen mit der in der Steuerung überein?
- Analyse mit Hilfe des Diagnose Viewers: Welche Meldungen werden angezeigt?