zenon
by COPA-DATA

zenon Treiber Handbuch

MtoM32

v.8.10

© 2019 Ing. Punzenberger COPA-DATA GmbH

Alle Rechte vorbehalten.

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

Inhaltsverzeichnis

1 Willkommen bei der COPA-DATA Hilfe 5

2 MtoM32 5

3 MtoM32 - Datenblatt 6

4 Treiber-Historie 7

5 Konfiguration 7

5.1 Anlegen eines Treibers 8

5.2 Einstellungen im Treiberdialog 11

5.2.1 Allgemein 12

5.2.2 Com 16

5.2.3 MtoM-Einstellungen 18

6 Variablen anlegen 31

6.1 Variablen im Editor anlegen 32

6.2 Adressierung 35

6.2.1 Anmerkungen zum Systemspeicher des PMI/PC 36

6.3 Treiberobjekte und Datentypen 37

6.3.1 Treiberobjekte 37

6.3.2 Zuordnung der Datentypen 37

6.4 Variablen anlegen durch Import 38

6.4.1 XML Import 39

6.4.2 DBF Import/Export 40

6.5 Kommunikationsdetails (Treibervariablen) 45

7 Treiberspezifische Funktionen 52

8 Funktion Treiberkommandos 52

9 Anschlusskabel 56

9.1 Anschlusskabel 1 (RS 232C) 57

9.2 Anschlusskabel 2 (RS 422) 57

9.3 Anschlusskabel 3 (RS 485) 58

10 Fehleranalyse 58

10.1 Analysetool 58

10.2 Checkliste 59

1 Willkommen bei der COPA-DATA Hilfe

ZENON VIDEO-TUTORIALS

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

ALLGEMEINE HILFE

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

PROJEKTUNTERSTÜTZUNG

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

LIZENZEN UND MODULE

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

2 MtoM32

Der **MtoM32**-Treiber ist ein passiver Treiber mit offengelegtem Protokoll für zenon.

Alle Schreib- und Lesezugriffe werden von dem angeschlossenen Kommunikationspartner/Host (SPS oder PC) gesteuert. Die externgesteuerte Bildschirmdarstellung des Leitsystems ändert sich nur nach einer Datenübertragung des Partners an zenon. Die Datenübertragung von zenon zum Master erfolgt durch einen Lesebefehl. Diesen Lesebefehl muss der Partner senden.

Ein Systemspeicher steht als Datenpuffer in zenon für die Kommunikation zur Verfügung.

Eine Verbindung ist auf PC-Systemen mit der Einstellung "RS 485" nicht möglich.

3 MtoM32 - Datenblatt

Allgemein:	
Treiberdateiname	MtoM32.exe
Treiberbezeichnung	Memory to Memory Treiber
Steuerungs-Typen	Beliebig. Protokoll muss implementiert sein.
Steuerungs-Hersteller	Pilz

Treiber unterstützt:	
Protokoll	Pilz MtoM
Adressierung: Adress-basiert	Address based
Adressierung: Namens-basiert	--
Kommunikation spontan	--
Kommunikation pollend	X
Online Browsing	--
Offline Browsing	--
Echtzeitfähig	--
Blockwrite	--
Modemfähig	--
RDA numerisch	--
RDA String	--
Hysterese	--
erweiterte API	--

Treiber unterstützt:	
Unterstützung von Statusbit **WR-SUC**	--
alternative IP-Adresse	--

Voraussetzungen:	
Hardware PC	--
Software PC	--
Hardware Steuerung	--
Software Steuerung	--
Benötigt v-dll	X

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

4 Treiber-Historie

Datum	Build Nummer	Änderung
20.09.2014	14584	Treiberdokumentation neu erstellt.

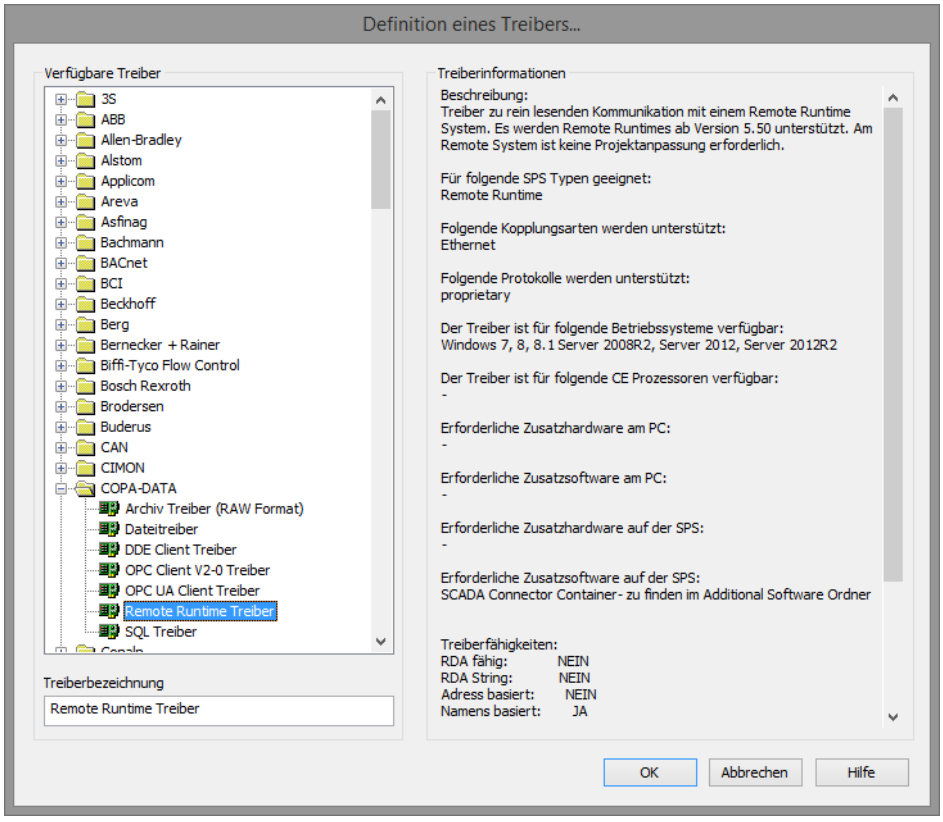
5 Konfiguration

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.

5.1 Anlegen eines Treibers

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



Parameter	Beschreibung
Verfügbare Treiber	Liste aller verfügbaren Treiber. Die Darstellung erfolgt in einer Baumstruktur: *[+]* erweitert die Ordnerstruktur und zeigt die darin enthaltenen Treiber. [-] reduziert die Ordnerstruktur Default: *keine Auswahl*
Treiberbezeichnung	Eindeutige **Bezeichnung** des Treibers. Default: *leer*

Parameter	Beschreibung
	Das Eingabefeld wird nach Auswahl eines Treibers aus der Liste der verfügbaren Treiber mit der vordefinierten **Bezeichnung** vorausgefüllt.
Treiberinformationen	Weiterführende Informationen über den gewählten Treiber. Default: *leer* Nach Auswahl eines Treibers werden in diesem Bereich die Informationen zum gewählten Treiber angezeigt.

DIALOG BEENDEN

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

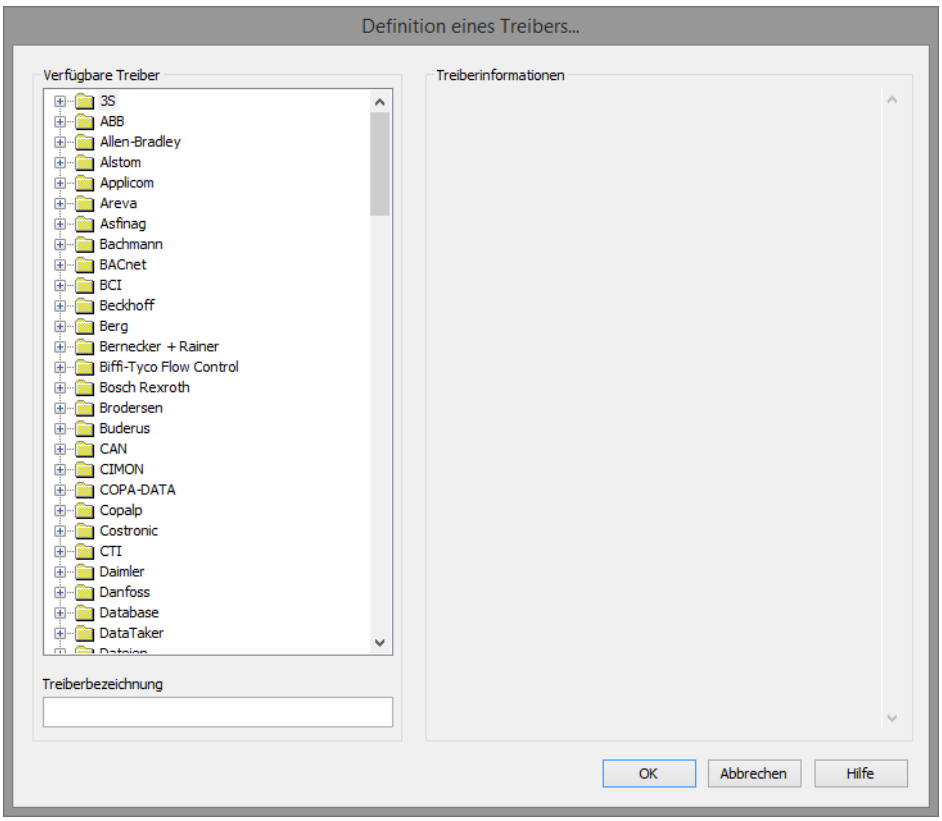
> **Info**
>
> Die Inhalte dieses Dialogs sind in der Datei Treiber_[Sprachkürzel].xml gespeichert. Sie finden diese Datei im Ordner *C:\ProgramData\COPA-DATA\zenon[Versionsnummer]*.

TREIBER NEU ANLEGEN

Um einen neuen Treiber anzulegen:

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

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



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

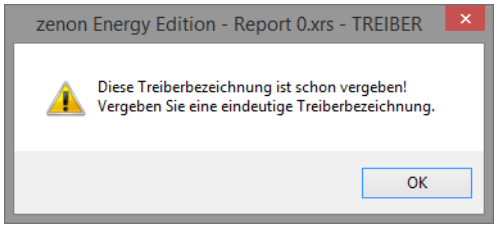
   - Die **Treiberbezeichnung** muss eindeutig sein.
     Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
     Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.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

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

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

DIALOG TREIBERBEZEICHNUNG BEREITS VORHANDEN

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



ZENON PROJEKT

Bei neu angelegten Projekten werden die folgenden Treiber automatisch angelegt:

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

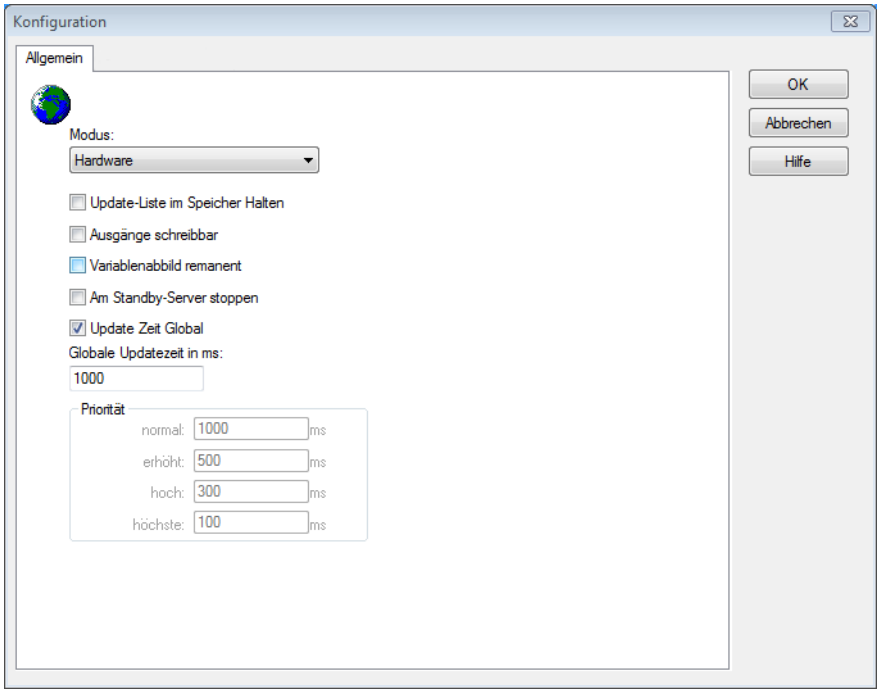
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

5.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

5.2.1 Allgemein

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



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

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

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

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

DIALOG BEENDEN

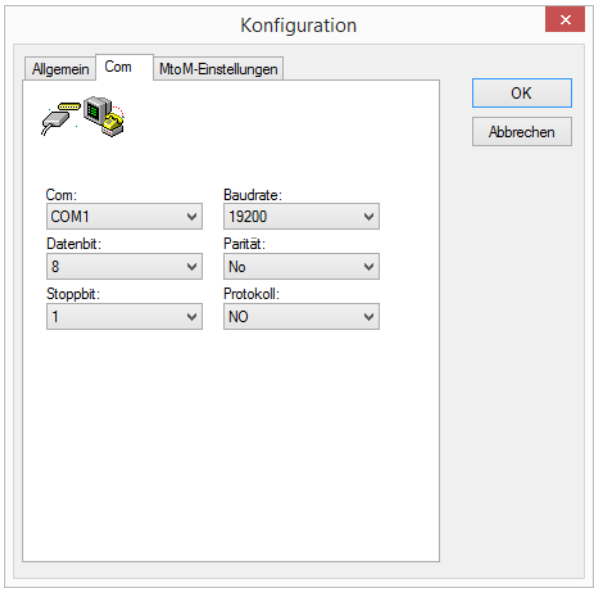
Option	Beschreibung
OK	**Übernimmt alle Änderungen in allen Registerkarten und schließt den Dialog.**
Abbrechen	Verwirft alle Änderungen in allen Registerkarten und schließt den Dialog.
Hilfe	Öffnet die Online-Hilfe.

UPDATE ZEIT SPONTANE TREIBER

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

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

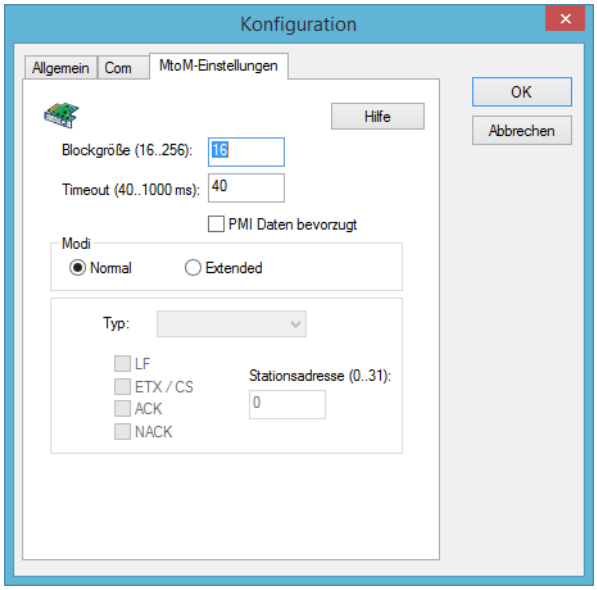
5.2.2 Com



Parameter	Beschreibung
Com:	Auswahl der seriellen Schnittstelle (physikalischer Kommunikationsport) am Rechner über welche die Kommunikation erfolgt: COM 1 bis COM 9 Default: *COM1*
Datenbit:	Dropdownliste zur Auswahl der Anzahl der Datenbits: ▸ 5 ▸ 6 ▸ 7 ▸ 8 Default: *8*
Stoppbit:	Dropdownliste zur Auswahl der Anzahl der Stopbits. Ermöglicht dem Empfänger eine Synchronisation auf jedes übertragene Zeichen.

Parameter	Beschreibung
	▸ 1 ▸ 1.5 ▸ 2 Default: *1*
Baudrate:	Dropdownliste zur Auswahl der Baudrate für die Kommunikation der Schnittstelle. Auswahlmöglichkeiten: *4800, 9600, 19200, 38400, 56000, 57600, 115200* Default: *19200*
Parität:	Auswahl des Paritätsprotokolles aus Dropdownliste: ▸ No Keine Parität - kein Kontrollbit ▸ Odd Paritätsbit wird bei gerader Anzahl Datenbits gesetzt. ▸ Even Paritätsbit wird bei ungerader Anzahl Datenbits gesetzt. Default: *No*
Protokoll:	Auswahl des Schnittstellenstandards für die Schnittstelle aus Dropdownliste: ▸ NO ▸ RS485 (für vergrößerte Störsicherheit) Default: *NO*

5.2.3 MtoM-Einstellungen



Parameter	Beschreibung
Blockgröße (16..256)	Interne Blockgröße zwischen Runtime und MtoM32-Treiber. Wertbereich: *16* bis *256* (Worte) Default: *16*
Timeout (40...1000 ms)	Zeichenverzugszeit. Timeoutzeit in Millisekunden Wenn innerhalb eines Telegrammes Pausen, länger als die Einstellung auftreten, wird das entsprechende Telegramm verworfen. Wertbereich: *40* bis *1000* Default: *40*
PMI Daten bevorzugt	Wenn aktiviert wird Multikast-Kommunikation vorrangig behandelt. Default: nicht aktiv
Modi	Dropdownliste zur Auswahl des Kommunikationsmodus. Diese Einstellung beeinflusst die weiteren Konfigurationsmöglichkeiten des Kommunikationstypes und der Stationsadresse.

Parameter	Beschreibung
	▸ Normal ▸ Extended Default: Normal Beachten Sie auch die Anmerkungen zum Systemspeicher (auf Seite 36)
Typ	Dropdownliste zur Auswahl des xx Typs: ▸ 1: 1 ASCII ▸ 1:1 Binary ▸ 1:N ASCII ▸ 1:N Binary Nur aktiv wenn Modi Extended ist Beachten Sie auch die Anmerkungen zum Systemspeicher (auf Seite 36)
zusätzliche Telegramme	Beim Extended Modus sind folgende Funktionen als Option wählbar: ▸ **LF** Line Feed **Hinweis:** nur aktiv wenn Typ ASCII ▸ **ETX / CS** End of Text/Checksumme ▸ **ACK** Acknowledge-Telegramm (ACK) ▸ **NACK** Not-Acknowledge-Telegramm (NAK) Die gewählten Funktionen werden an das Antwort- bzw. Sendetelegramm angehängt
Stationsadresse (0...31):	Bei den *1:n Modi* können mehrere Stationen (Slave) an den Host (Master) angeschlossen werden. Die Eingabe einer Stationsadresse ist in diesem Falle zwingend notwendig. Wertbereich: *0 bis 31 (00_{Dec} bis $1F_{Hex}$)* Default: *0*

Parameter	Beschreibung
	Nur aktiv wenn Typ *1:N* ist. **Hinweis:** Wird als Stationsadresse 255_{Dec} (FF^{Hex}) angegeben, wird das Telegramm von allen Leitsystemen empfangen. Wird als Stationsnummer 255_{Dec} (FF_{Hex}) angegeben, so wird das Telegramm von allen PMI empfangen.

> **Achtung**
>
> Auf ein Telegramm mit der Adresse 255_{Dec} (FF_{Hex}) folgt nie ein Antwort-telegramm - auch kein ACK oder NAK.

5.2.3.1 Kommunikationsmodi

Insgesamt stellt der **MtoM32**-Treiber fünf verschiedene Kommunikationsmodi zur Verfügung:

- Normalmodus (ASCII Codes)
- Extended Modus
  - 1:1 ASCII Modus
  - 1:1 Binary Modus
  - 1:n ASCII Modus
  - 1:n Binary Modus

Der *Normalmodus* und die *Extended Modi 1:1 ASCII* oder *1:1 Binary* sind für Punkt zu Punkt Kopplungen geeignet. Dabei sind ein Master (Steuerung, Microcontroller, PC) und ein Leitsystem/PC vorhanden. Der Master sendet Lese- und Schreibtelegramme an das Leitsystem/den PC.

Die *Extended Modi 1:n ASCII* oder *1:n Binary* ermöglichen den Anschluss mehrerer Leitsysteme (Slaves) an einen Master. Der Master kommuniziert dann abwechselnd mit den angeschlossenen Geräten. Die Reihenfolge, wann mit welchem Slave kommuniziert wird, bestimmt das Anwenderprogramm im Master. Auch die Möglichkeit eines Sendetelegrammes an alle Slaves (Sammelaufruf) besteht, jedoch nur beim Schreibbefehl. Damit lassen sich Daten, die alle Slaves benötigen sehr schnell übertragen. Beim Sammelaufruf antworten die angeschlossenen Slaves nicht. Es gibt also keine Rückinformation, ob das Telegramm verstanden wurde oder nicht.

UNTERSCHIED ASCII-CODE - BINARY-CODE

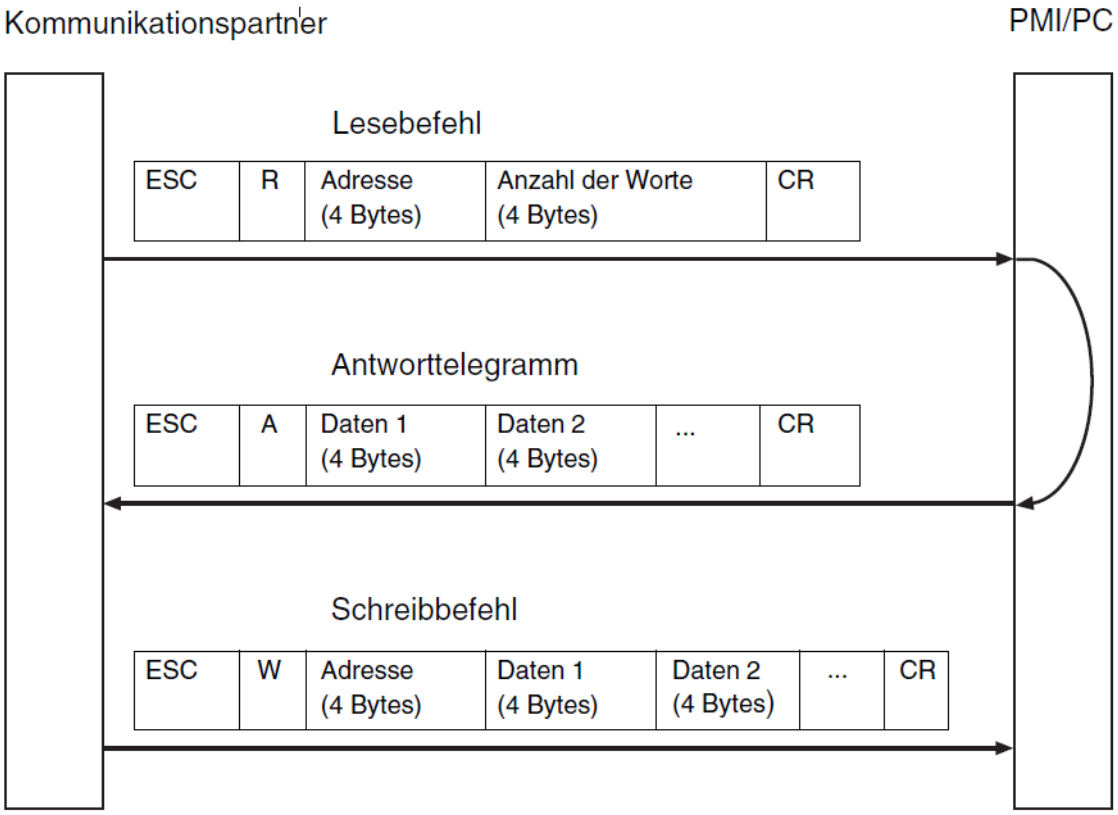
Bei den Modi *Normal*, *1:1 ASCII* und *1:n ASCII* werden alle Zeichen in ASCII Zeichen umgewandelt. Die Steuerzeichen sind im Hex (Binary)-Code organisiert. Dadurch sind die Daten im Telegramm eindeutig von den Steuerzeichen zu unterscheiden.

Die Binary Modi (*1:1 Binary* und *1:n Binary*) übertragen die Steuerzeichen und die Daten im Binär(Hex)-Code. Hierdurch entsteht eine Art von Datenkomprimierung.

Info

Im Binary-Modus können Steuerzeichen auch in den Nutzdaten vorkommen, die wie Ende-Kennungen (ETX, CR, LF) aussehen. Der Master hat somit kein eindeutiges Ende-Zeichen für ein Telegramm

5.2.3.1.1 Normalmodus



Zeichen	Bezeichnung	Bedeutung	Hex
R	*Read*	Lesen von Daten aus dem PMI/PC	52
A	*Answer*	Antwort auf Lesetelegramm	41
W	*Write*	Schreiben von Daten in PMI/PC	57
CR	*Carrige Return*	Endzeichen Kommando	0D
ESC	*Escape*	Startzeichen Kommando	1b

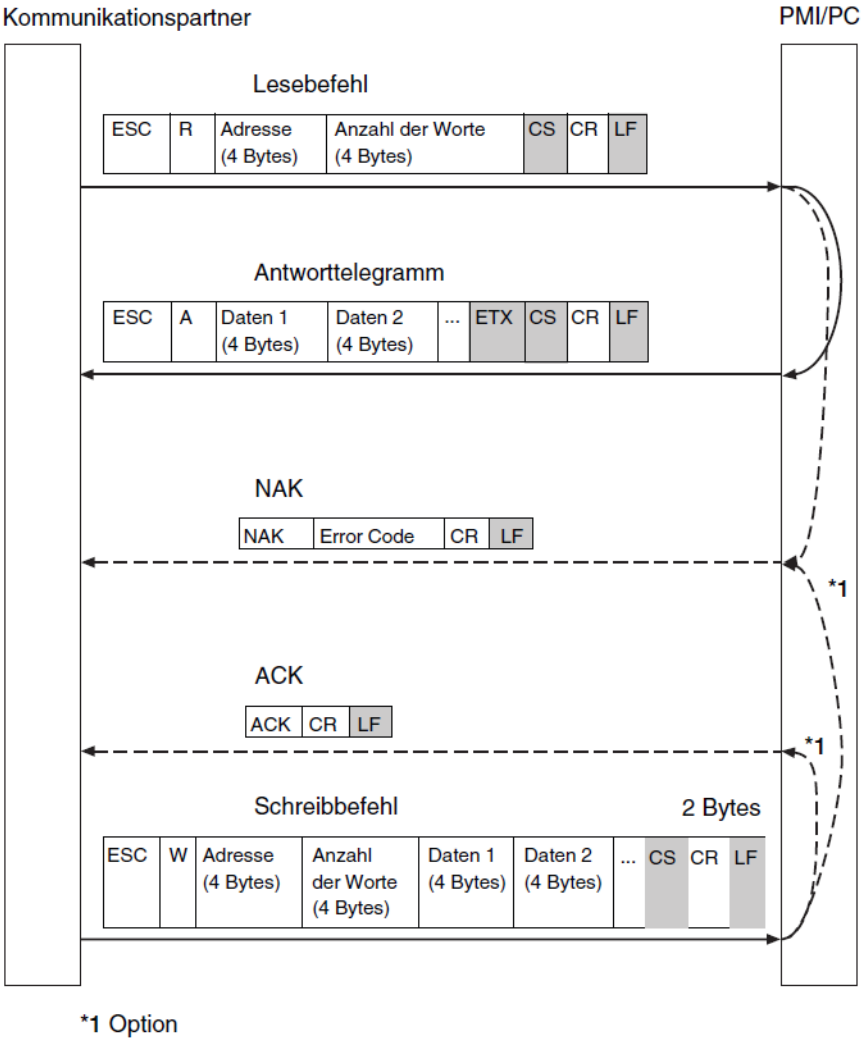
SCHREIBBEFEHL

Mit einem Schreibbefehl werden Daten von der Steuerung zum Slave (PMI/PC) übertragen. Der Startcode, der Schreibbefehl und das Return werden hexadezimal eingegeben. Die Adresse und die Daten werden im ASCII-Code eingegeben.

5.2.3.1.2 Extended Modus

Optionale Steuerzeichen werden in den folgenden Darstellungen grau unterlegt. Optionale Telegramme werden mit gestrichelten Pfeillinien gekennzeichnet

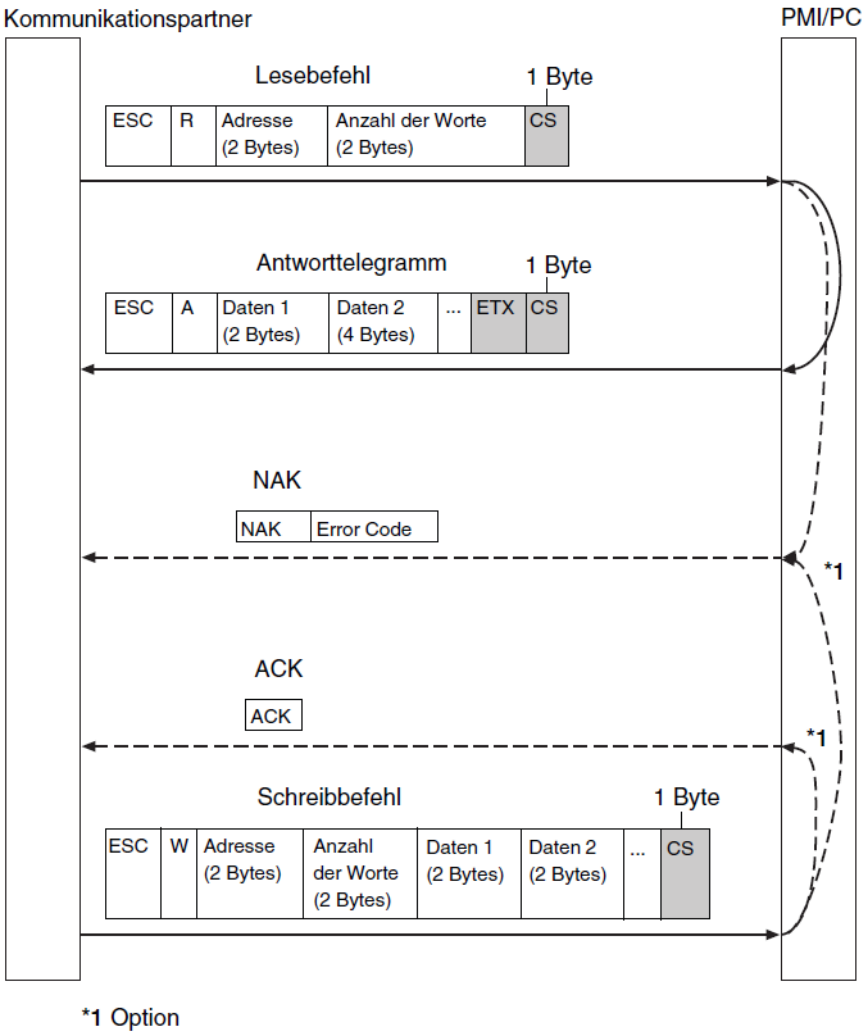
5.2.3.1.31:1 ASCII Modus



Zeichen	Bezeichnung	Bedeutung	Hex
R	Read	Lesen von Daten aus dem PMI/PC	52
A	Answer	Antwort auf Lesetelegramm	41
W	Write	Schreiben von Daten in PMI/PC	57
ACK	Acknowledge	Daten empfangen ohne Fehler	06
NAK	Negativ Acknowledge	Fehler beim Datenempfang	15
CR	Carrige Return	Endezeichen Kommando	0D
LF	Line Feed	Endezeichen Kommando	0A

Zeichen	Bezeichnung	Bedeutung	Hex
ETX	End of Text	Endezeichen Telegramm	03
ESC	Escape	Startzeichen Telegramm	1B
CS	Checksum	Checksumme	-

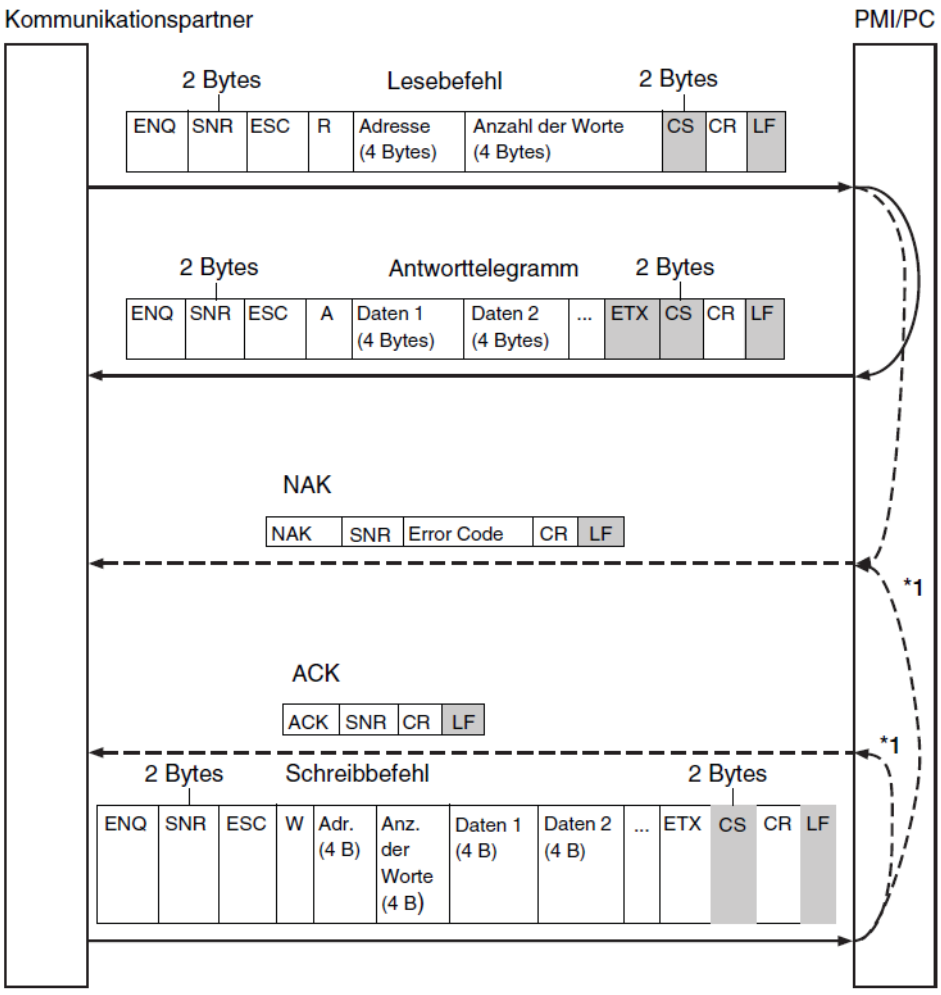
5.2.3.1.41:1 Binary Modus



Zeichen	Bezeichnung	Bedeutung	Hex
R	Read	Lesen von Daten aus dem PMI/PC	52
A	Answer	Antwort auf Lesetelegramm	41

Zeichen	Bezeichnung	Bedeutung	Hex
W	Write	Schreiben von Daten in PMI/PC	57
ACK	Acknowledge	Daten empfangen ohne Fehler	06
NAK	Negativ Acknowledge	Fehler beim Datenempfang	15
ETX	End of Text	Endezeichen Telegramm	03
ESC	Escape	Startzeichen Kommando	1B
CS	Checksum	Checksumme	-

5.2.3.1.51:n ASCII Modus



*1 Option

Zeichen	Bezeichnung	Bedeutung	Hex
R	Read	Lesen von Daten aus dem PMI/PC	52
A	Answer	Antwort auf Lesetelegramm	41
W	Write	Schreiben von Daten in PMI/PC	57
ACK	Acknowledge	Daten empfangen ohne Fehler	06
NAK	Negativ Acknowledge	Fehler beim Datenempfang	15
CR	Carrige Return	Endezeichen Kommando	0D
LF	Line Feed	Endezeichen Kommando	0A
ETX	End of Text	Endezeichen Telegramm	03
ENQ	Enquiry	Startzeichen Telegramm	05
ESC	Escape	Startzeichen Kommando	1B
SNR	Slave Number	Slave Nummer	-
CS	Checksum	Checksumme	-

Bei den 1:n Modi können mehrere Stationen (Slave) an den Host (Master) angeschlossen werden. Deswegen muss in den Telegrammen zusätzlich die Stationsnummer angegeben werden.

Hinweis: Wird als Stationsnummer 255_{Dec} (FF_{Hex}) angegeben, so wird das Telegramm von allen Leitsystemen empfangen.

Info

Auf ein Telegramm mit der Adresse 255_{Dec} (FF_{Hex}) folgt nie ein Antworttelegramm - auch kein ACK oder NAK

LESEBEFEHL IM 1:N ASCII MODUS (MIT ETX/CS MIT CR + LF, OHNE ACK, OHNE NAK)

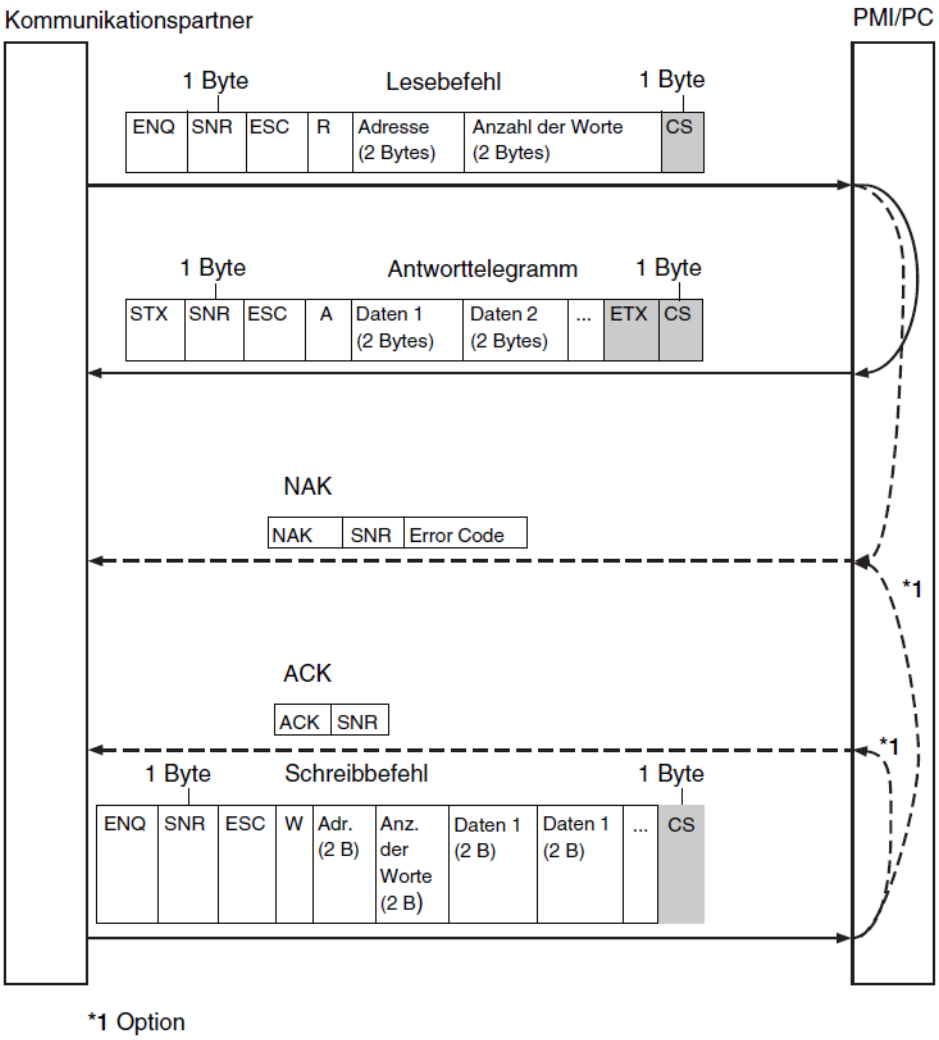
Das Leitsystem mit der Teilnehmeradresse 02 soll den Inhalt der Merker-Wörter 101, 102, 103, 104 und 105 an den Partner senden.

ENQ	SNR 2 Bytes	ESC	R	Adresse (4 Bytes)	Anzahl der Worte (4 Bytes)	CS 2 Bytes	CR	LF
05	30 32	1B	52	30 30 36 35	30 30 30 35	45 32	0D	0A

Inhalt von	Wert (Dec)		Wert (Hex)
MW 101:	12345	3039_{Hex}	33 30 33 39
MW 102:	6789	$1A85_{Hex}$	31 41 38 35
MW103:	2	0002_{Hex}	30 30 30 32
MW 104:	24671	$605F_{Hex}$	36 30 35 46
MW 105	13572	3504_{Hex}	33 35 30 34

ENQ	SNR 2 Bytes	ESC	A	MW 101	MW 102	MW 103	MW 104	MW 105	ETX	CS 2 Bytes	CR	LF
05	30 32	1B	41	33 30 33 39	31 41 38 35	30 30 30 32	36 30 35 46	33 35 30 34	03	45 32	0D	0A

5.2.3.1.61:n Binary Modus



*1 Option

Zeichen	Bezeichnung	Bedeutung	Hex
R	Read	Lesen von Daten aus dem PMI/PC	52 H
A	Answer	Antwort auf Lesetelegramm	41 H
W	Write	Schreiben von Daten in PMI/PC	57
ACK	Acknowledge	Daten empfangen ohne Fehler	06 H

Zeichen	Bezeichnung	Bedeutung	Hex
NAK	Negativ Acknowledge	Fehler beim Datenempfang	15 H
ETX	End of Text	Endezeichen Telegramm	03 H
STX	Start of Text	Startzeichen Telegramm	02 H
ENQ	Enquiry	Startzeichen Telegramm	05 H
ESC	Escape	Startzeichen Kommando	1B H
SNR	Slave Number	Slave Nummer	-
CS	Checksum	Checksumme	-

LESEBEFEHL IM 1:N BINÄRY MODUS (MIT ETX/CS MIT CR + LF, OHNE ACK, OHNE NAK)

Das PMI/der PC mit der Teilnehmadresse 02 soll den Inhalt der Wörter 101, 102, 103, 104 und 105 an den Partner senden.
Der Partner sendet folgende Lesebefehle:

ENQ	SNR 1 Byte	ESC	R	Adresse (4 Bytes)	Anzahl der Worte (4 Bytes)	CS 1 Byte
05	02	1B	52	30 30 36 35	30 30 30 35	38

ANTWORTTELEGRAMM IM 1:N BINÄRY MODUS

Das Leitsystem antwortet mit dem Antworttelegramm

Inhalt von	Wert (Dec)	Wert (Hex)
MW 101:	12345	33 30 33 39
MW 102:	6789	31 41 38 35
MW103:	2	30 30 30 32

Inhalt von	Wert (Dec)	Wert (Hex)
MW 104:	24671	36 30 35 46
MW 105	13572	33 35 30 34

STX	SNR 1 Byte	ESC	A	MW 101	MW 102	MW 103	MW 104	MW 105	ETX	CS 1 Byte
05	02	1B	41	33 30 33 39	33 41 38 35	30 30 30 32	33 36 35 46	33 35 30 34	03	63

5.2.3.2 Berechnung der Checksumme

Die Checksumme dient der Prüfung des Telegramms auf korrekte Übertragung. Durch Störeinflüsse können die übertragenen Daten verfälscht werden. Dies lässt sich teilweise mit der Checksumme erkennen.

Grundsätzlich stellt die Checksumme die Summe aller übertragenen Daten dar.

Info

Da für die Checksumme ein Byte vorgesehen ist, kann diese maximal den Wert 255_{Dec} (FF_{Hex}) annehmen. Übersteigt die berechnete Summe diesen Wert, werden nur die zwei letzten Ziffern der Summe in hexadezimaler Darstellung als Checksumme übernommen.
Tipp: Beachten Sie Dazu das folgende Beispiel 2.

Bei den ASCII Modi müssen diese zwei Ziffern zusätzlich in ASCII-Zeichen umgewandelt werden. Die Startzeichen *ENQ* und *STX* werden nicht in die Checksumme eingerechnet. Alle Zeichen nach der Checksumme werden ebenso nicht eingerechnet.

5.2.3.2.1 Beispiele zur Berechnung der Checksumme:

BEISPIEL 1 - MODUS 1:N ASCII

Für Modus 1:n ASCII (mit ETX/CS, mit CR+LF)

Schreiben des Wertes 00C8Hex (200D) an Adresse 0064Hex

(100Dec) auf Slave 01

ENQ	SNR 2 Bytes	ESC	W	MW 100	MW 200	CS 2 Bytes	CR	LF
05	30 31	1B	57	30 30 36 34	30 30 43 38	37 38	0D	0A
	Bereich zur Berechnung der Checksumme							

CS

= 30+31+1B+57+30+30+36+34+30+30+43+38

= 278Hex --> 78Hex --> ASCII: 37 38

BEISPIEL 2 - MODUS 1:N BINARY

Für Modus 1:n Binary (mit ETX/CS, ohne CR+LF)

Antwort vom PMI/PC auf Lesetelegramm, Rückgabe der Werte 1234Hex

(4660Dec) und 4321Hex (17185 dec) von Slave 12Hex (18Dec)

STX	SNR 2 Bytes	ESC	A	Wert 4660	Wert 17185	ETX	CS 1 Byte
02	12	1B	41	12 34	43 21	03	1 B
	Bereich zur Berechnung der Checksumme						

CS

= 12+1B+41+12+34+43+43+21+03 = 11BHex

= 11BHex --> nur letztes Byte: 1 BHex

6 Variablen anlegen

So werden Variablen im zenon Editor angelegt:

6.1 Variablen im Editor anlegen

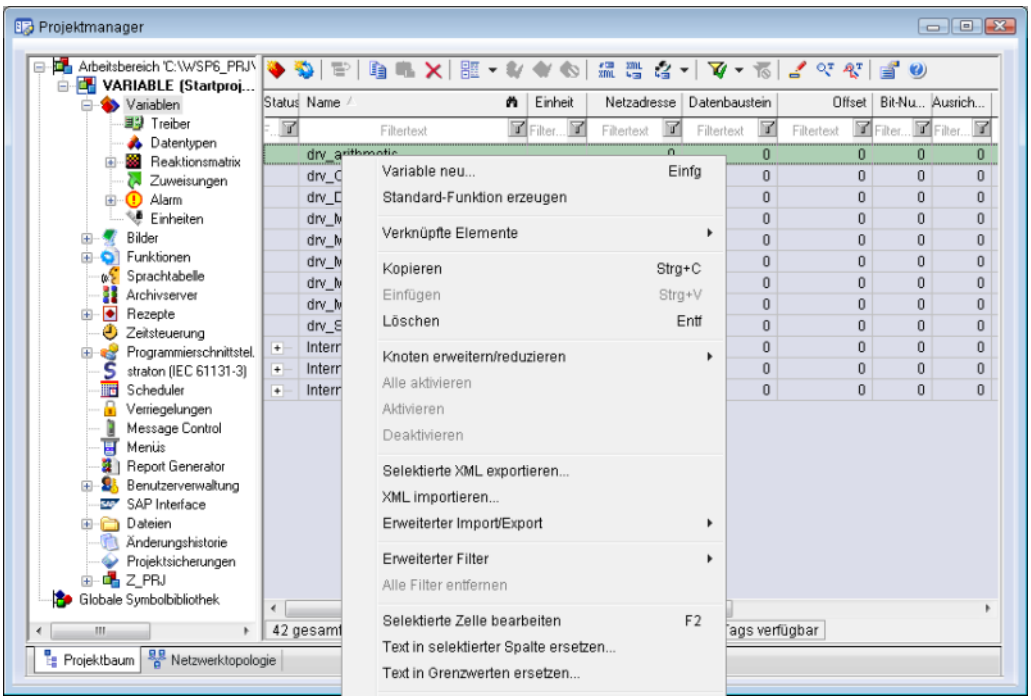
Variablen können angelegt werden:

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

DIALOG VARIABLE

Um eine neue Variable zu erstellen, gleich welchen Typs:

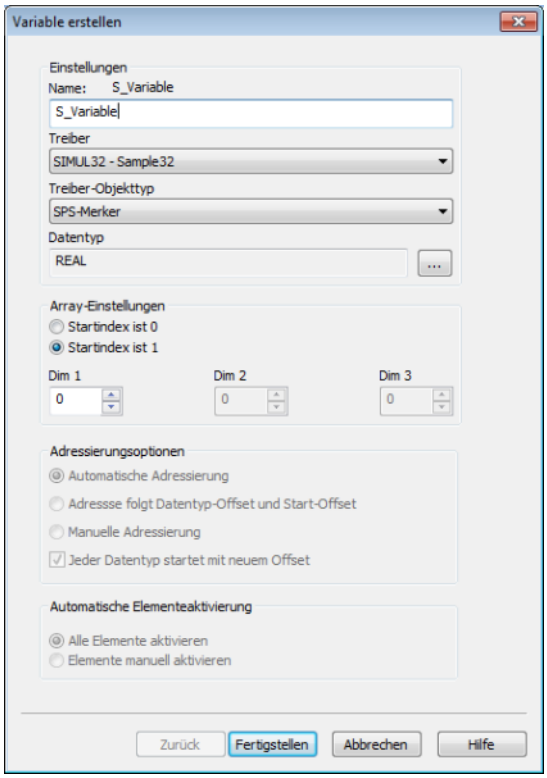
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



   Der Dialog zur Konfiguration der Variable wird geöffnet.

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

DIALOG VARIABLE ERSTELLEN



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

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

SYMBOLISCHE ADRESSE

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

Maximale Länge: *1024* Zeichen.

Folgende Treiber unterstützen die **Symbolische Adresse**:

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

ABLEITUNG VOM DATENTYP

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

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

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

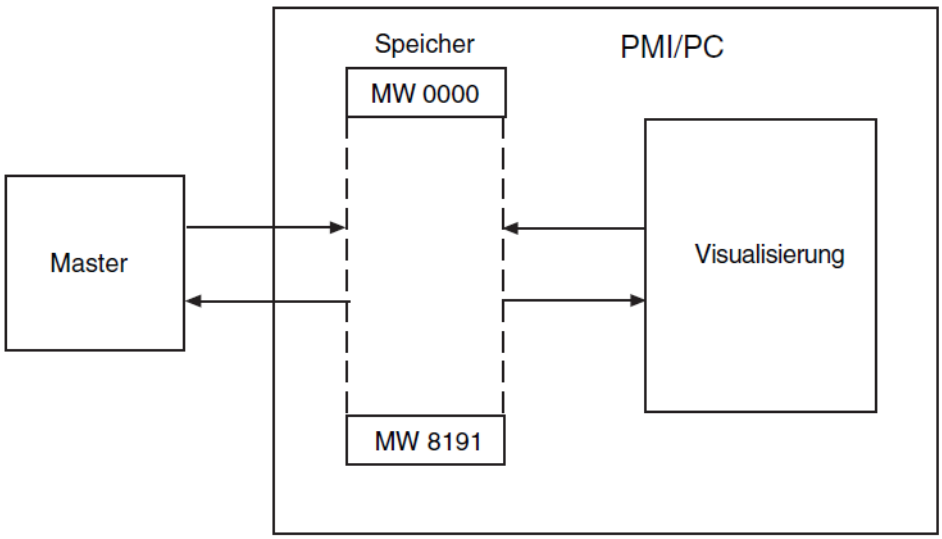
6.2 Adressierung

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	Wird für diesen Treiber nicht verwendet.
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Offset der Variablen, die Speicheradresse der Variable in der Steuerung. Mögliche Eingabe: *0* bis *8191*. MtoM32 spezifische Info: - Der Systemspeicher ist **wort-orientiert** - Jedes Wort hat ein niederwertiges Byte (Lowbyte) und ein höherwertiges Byte (Highbyte). - Jedes Wort hat die Bits 0 bis 15 **Achtung:** - Das *Merkerwort 13* ist für weitere Entwicklungen reserviert! - Beachten Sie auch die Anmerkungen zum Systemspeicher (auf Seite 36)
Ausrichtung	Wird für diesen Treiber nicht verwendet.
Bitnummer	Nummer des Bits innerhalb des eingestellten Offsets.

Gruppe/Eigenschaft	Beschreibung
	Mögliche Eingabe: *0 ... 15*
Stringlänge	Nur verfügbar bei String-Variablen. Maximale Anzahl von Zeichen, die die Variable aufnehmen kann. **Achtung:** Höchster Offset für String-Merker ist *8159*
Treiber Anbindung/Treiber objekttyp	Objekttyp der Variablen. Wird abhängig vom verwendeten Treiber beim Erstellen der Variablen ausgewählt und kann hier geändert werden.
Treiber Anbindung/Datentyp	Datentyp der Variablen. Wird beim Erstellen der Variablen ausgewählt und kann hier geändert werden. **ACHTUNG:** Wenn der Datentyp nachträglich geändert wird, müssen alle anderen Eigenschaften der Variablen überprüft bzw. angepasst werden.

6.2.1 Anmerkungen zum Systemspeicher des PMI/PC

Der Systemspeicher dient als Schnittstelle zur Datenübertragung zwischen dem Leitsystem und seinem Partner. Auf diesen Speicher wird von beiden Seiten (Partner/Visualisierung) geschrieben oder gelesen. Abhängig von den Daten im Systemspeicher ändert das Leitsystem die Anzeige. Der Speicher hat eine Größe von 8192 Worten. Darüber hinaus gehende Adressen dürfen nicht verwendet werden.



BESCHREIBUNG DER SPEICHERZUGRIFFE

zenon liest eigenständig die Daten des Systemspeicher aus. Der Partner bestimmt, wann er Daten benötigt und fordert diese mit einem Befehl an. Daraufhin schreibt zenon die angeforderten Daten in den Systemspeicher.

- Lesebefehl:
  Der Partner sendet eine Leseanforderung zum Leitsystem.
- Antworttelegramm:
  zenon/der PC antwortet auf den Lesebefehl und sendet die angeforderten Daten zur Steuerung.
- Schreibtelegramm:
  Die Steuerung sendet Daten an zenon/den PC in den Systemspeicher.

6.3 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.

6.3.1 Treiberobjekte

Folgende Objekttypen stehen in diesem Treiber zur Verfügung:

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Beschreibung
SPS-Merker	80	X	X	*BOOL*, *SINT*, *USINT*, *INT*, *UINT*, *DINT*, *UDINT*, *REAL*, *STRING*	Adressierung erfolgt wort-orientiert. UDINT ist zwei Wörter lang und beginnt jeweils bei jedem zweiten Speicherwort**:** 0 2 4 6 8 ...

Legende:

X: wird unterstützt

--: wird nicht unterstützt

6.3.2 Zuordnung der Datentypen

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

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

DATENART

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

6.4 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).

6.4.1 XML Import

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

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

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

VORAUSSETZUNGEN

Beim Import gelten folgende Bedingungen:

- **Abwärtskompatibilität**

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

- **Konsistenz**

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

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

- **Struktur-Datentypen**

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

Tipp

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

6.4.2 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

Bezeichnung	Typ	Feldgröße	Bemerkung
			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

Bezeichnung	Typ	Feldgröße	Bemerkung
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)

Bezeichnung	Typ	Feldgröße	Bemerkung
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.

6.5 Kommunikationsdetails (Treibervariablen)

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

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

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

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

Info

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

Zum Beispiel:

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

INFORMATION

Name aus Import	Typ	Offset	Erklärung
MainVersion	*UINT*	0	Haupt-Versionsnummer des Treibers.
SubVersion	*UINT*	1	Sub-Versionsnummer des Treibers.
BuildVersion	*UINT*	29	Build-Versionsnummer des Treibers.
RTMajor	*UINT*	49	zenon Hauptversionsnummer
RTMinor	*UINT*	50	zenon Sub-Versionsnummer
RTSp	*UINT*	51	zenon Service Pack-Nummer
RTBuild	*UINT*	52	zenon Buildnummer
LineStateIdle	*BOOL*	24.0	TRUE, wenn die Modemleitung belegt ist.

Name aus Import	Typ	Offset	Erklärung
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

Name aus Import	Typ	Offset	Erklärung
			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.

Name aus Import	Typ	Offset	Erklärung
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).

Name aus Import	Typ	Offset	Erklärung
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

Name aus Import	Typ	Offset	Erklärung
			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.

7 Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

8 Funktion Treiberkommandos

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

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

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

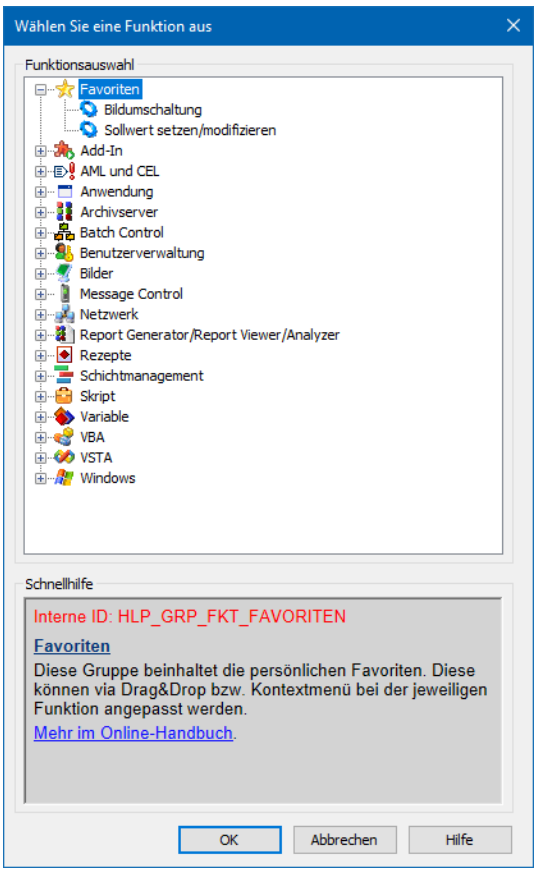
> **Achtung**
>
> Die zenon Funktion **Treiberkommandos** ist nicht ident mit den Treiberkommandos, die bei Energy-Treibern in der Runtime ausgeführt werden können!

PROJEKTIERUNG DER FUNKTION

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

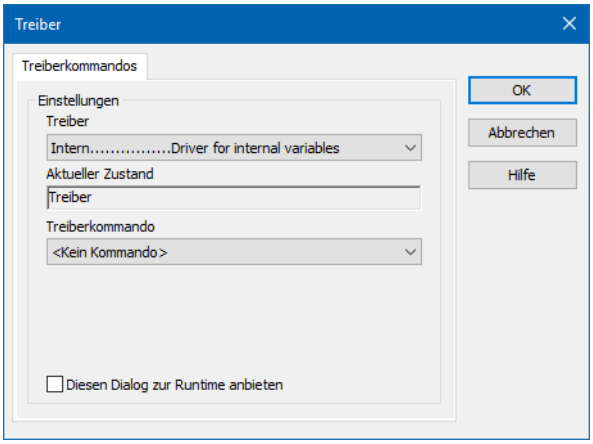
1. Legen Sie im zenon Editor eine neue Funktion an.

Der Dialog zur Auswahl einer Funktion wird geöffnet.



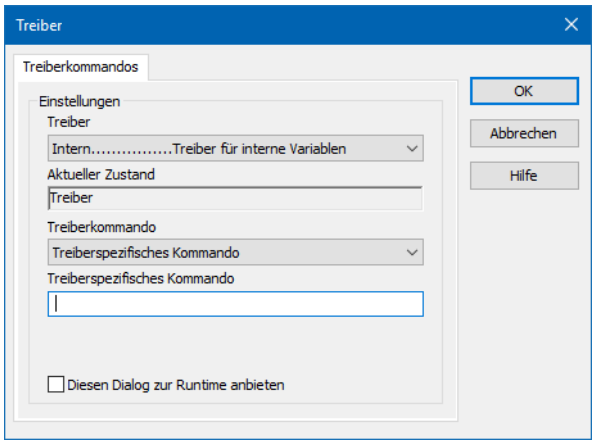
2. Navigieren Sie zum Knoten **Variable.**
3. Wählen Sie den Eintrag **Treiberkommandos**.

   Der Dialog zur Konfiguration wird geöffnet.



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

DIALOG TREIBERKOMMANDOS



Option	Beschreibung
Treiber	Auswahl des Treibers aus der Dropdownliste. Diese enthält alle im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, der vom System gesetzt wird. In aktuellen Versionen ohne Funktion.
Treiberkommando	Auswahl des gewünschten Treiberkommandos aus Dropdownliste. Details zu den konfigurierbaren Treiberkommandos siehe Abschnitt **Verfügbare Treiberkommandos**.
Treiberspezifisches Kommando	Eingabe eines für den gewählten Treiber spezifischen Kommandos. **Hinweis:** Nur verfügbar, wenn für die Option **Treiberkommando** der Eintrag *Treiberspezifisches Kommando* gewählt wurde.
Diesen Dialog zur Runtime anbieten	Konfiguration, ob die Konfiguration in der Runtime geändert werden kann: ▸ *Aktiv*: Dieser Dialog wird in der Runtime vor dem Ausführen der Funktion geöffnet. Die Konfiguration kann damit in der Runtime vor der Ausführung noch geändert werden. ▸ *Inaktiv*: Die Editor-Konfiguration wird in der Runtime beim Ausführen der Funktion angewendet. Default: *inaktiv*

DIALOG BEENDEN

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

VERFÜGBARE TREIBERKOMMANDOS

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

Treiberkommando	Beschreibung
<Kein Kommando>	Es wird kein Kommando gesendet. Damit kann auch ein bereits bestehendes Kommando aus einer projektierten Funktion entfernt werden.
Treiber starten (Onlinemodus)	Treiber wird neu initialisiert und gestartet. **Hinweis:** Wenn der Treiber bereits gestartet wurde, muss er gestoppt werden. Erst dann kann der Treiber wieder neu initialisiert und gestartet werden.
Treiber stoppen (Offlinemodus)	Treiber wird angehalten, es werden keine neuen Daten angenommen. **Hinweis:** Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status *Abgeschaltet* (*OFF*; Bit *20*).
Treiber in Simulationsmodus	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiber in Hardwaremodus	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
Treiberspezifisches Kommando	Eingabe eines treiberspezifischen Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist möglich.
Treiber Sollwertsetzen deaktivieren	Sollwert setzen auf Treiber wird verhindert.

Treiberkommando	Beschreibung
Verbindung mit Modem aufbauen	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
Verbindung mit Modem trennen	Verbindung beenden (für Modem-Treiber).
Treiber in Simulationsmodus zählend	Treiber wird in den zählenden Simulationsmodus gesetzt. Alle Werte werden mit *0* initialisiert und in der eingestellten Updatezeit jeweils um *1* bis zum Maximalwert inkrementiert und beginnen dann wieder bei *0*.
Treiber in Simulationsmodus statisch	Es wird keine Kommunikation zur Steuerung aufgebaut. Alle Werte werden mit *0* initialisiert.
Treiber in Simulationsmodus programmiert	Die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in der zenon Logic Runtime ab.

FUNKTION TREIBERKOMMANDOS IM NETZWERK

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

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

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

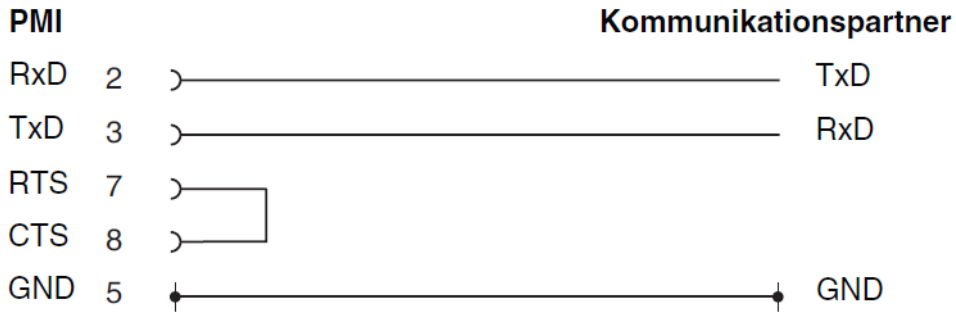
9 Anschlusskabel

Übersicht der Pin-Belegung für folgende Anschlusskabel:

- Anschlusskabel 1 (RS 232C) (auf Seite 57)
- Anschlusskabel 2 (RS 422) (auf Seite 57)
- Anschlusskabel 3 (RS 485) (auf Seite 58)

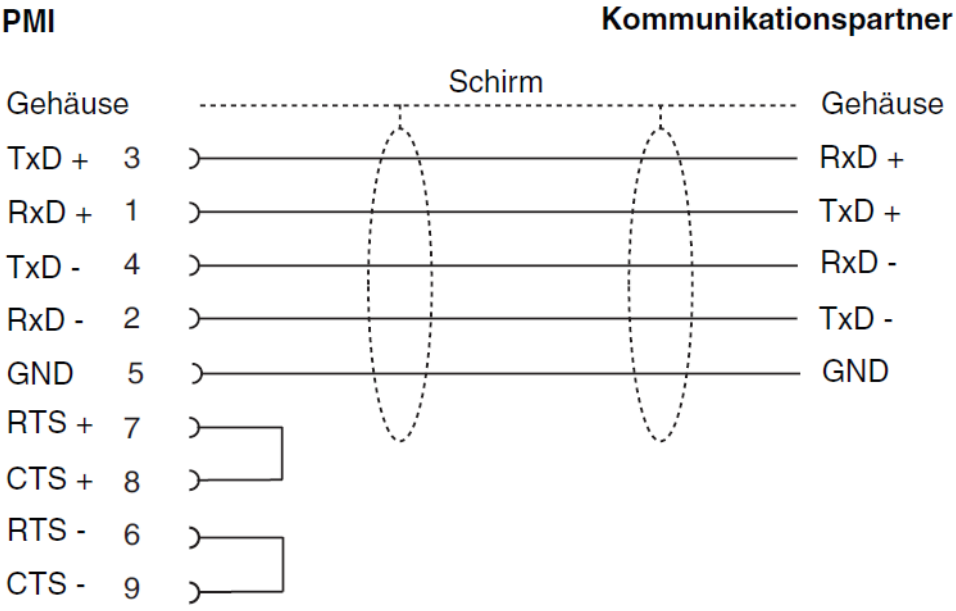
9.1 Anschlusskabel 1 (RS 232C)

PIN-Belegung Anschlusskabel 1 (RS 232C)



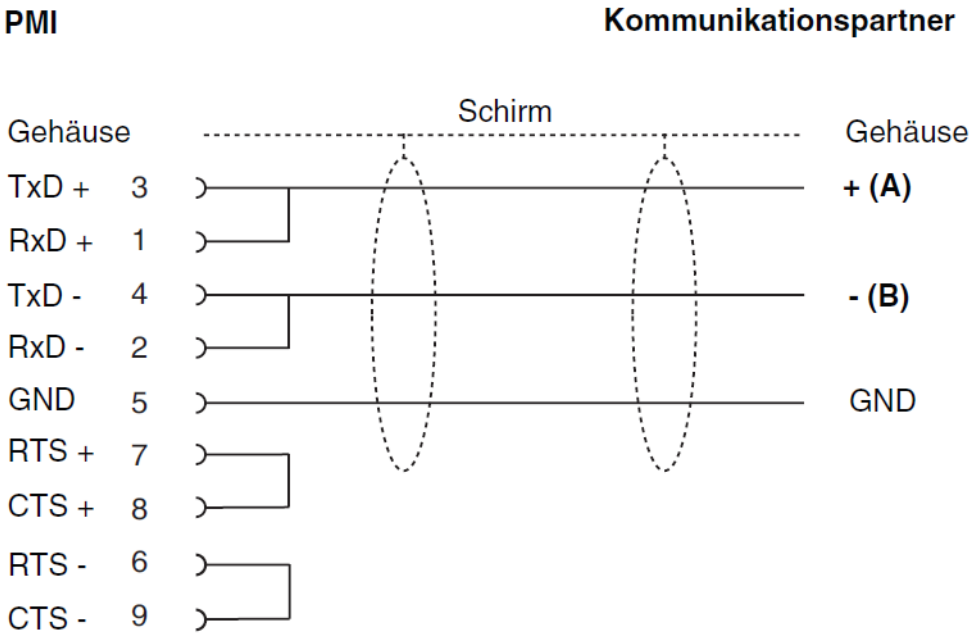
9.2 Anschlusskabel 2 (RS 422)

PIN-Belegung Anschlusskabel 1 (RS 422)



9.3 Anschlusskabel 3 (RS 485)

PIN-Belegung Anschlusskabel 1 (RS 485)



10 Fehleranalyse

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

10.1 Analysetool

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

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

%ProgramData%\COPA-DATA\LOG.

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

Im Diagnosis Viewer kann man auch:

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

Hinweise:

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

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

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

10.2 Checkliste

Fragen und Hinweise zur Fehlereingrenzung:

ALLGEMEINE FEHLERSUCHE

- Ist die Steuerung an die Stromversorgung angeschlossen?
- Analyse mit Hilfe des Diagnosis Viewers (auf Seite 58):
  -> Welche Meldungen werden angezeigt?

- Wird für die Verbindung von Steuerung und PC das korrekte, vom Hersteller empfohlene Kabel verwendet?
- Wurde der richtige COM Port ausgewählt?
- Stimmen die seriellen Kommunikationsparameter (Baudrate, Parität, Start/Stop Bits,...) überein?
- Wird der COM Port durch eine andere Anwendung blockiert?
- Wird in der Variable der richtige Objekttyp verwendet?
  Beispiel: Variablen vom Treiberobjekttyp **Kommunikationsdetails** sind reine Statistikvariablen und kommunizieren nicht mit der Steuerung.
  Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 45).
- Stimmt die Offset-Adressierung der Variable mit der in der Steuerung überein?

WERTE WERDEN NICHT ANGEZEIGT, ZAHLENWERTE BLEIBEN LEER

Treiber läuft nicht. Überprüfen Sie die:

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

VARIABLEN WERDEN MIT BLAUEM PUNKT ANGEZEIGT

Die Kommunikation im Netzwerk ist gestört:

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

WERTE WERDEN FALSCH ANGEZEIGT

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

TREIBER FÄLLT SPORADISCH AUS

Analyse mit Hilfe des Diagnosis Viewers (auf Seite 58):
-> Welche Meldungen werden angezeigt?