zenon Treiber Handbuch

BrTcp32

v.7.50

©2016 Ing. Punzenberger COPA-DATA GmbH

Alle Rechte vorbehalten.

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

Inhalt

1. Willkommen bei der COPA-DATA Hilfe 5

2. BrTcp32 6

3. BRTCP32 - Datenblatt 6

4. Treiber-Historie 7

5. Voraussetzungen 8

5.1 PC 8

5.2 Steuerung 9

6. Konfiguration 9

6.1 Anlegen eines Treibers 9

6.2 Einstellungen im Treiberdialog 11

6.2.1 Allgemein 11

6.2.2 Basic settings 15

6.2.3 Conn. TCP/IP 16

7. Variablen anlegen 17

7.1 Variablen im Editor anlegen 17

7.2 Adressierung 21

7.3 Treiberobjekte und Datentypen 22

7.3.1 Treiberobjekte 22

7.3.2 Zuordnung der Datentypen 24

7.4 Variablen anlegen durch Import 24

7.5 Treibervariablen 24

8. Treiberspezifische Funktionen 31

9. Treiberkommandos 32

10. Fehleranalyse 34

10.1 Analysetool 34

10.2 Fehlernummern 36

10.3 Checkliste 45

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. BrTcp32

3. BRTCP32 - Datenblatt

Allgemein:	
Treiberdateiname	BRTCP32.exe
Treiberbezeichnung	BuR TCP-IP
Steuerungs-Typen	B+R System 2000 und B+R Burklimat
Steuerungs-Hersteller	Bernecker + Rainer;

Treiber unterstützt:	
Protokoll	B+R TCP;
Adressierung: Adress-basiert	X
Adressierung: Namens-basiert	--
Kommunikation spontan	X
Kommunikation pollend	--
Online Browsing	--
Offline Browsing	--
Echtzeitfähig	X
Blockwrite	--
Modemfähig	--
Serielles Logging	--
RDA numerisch	--
RDA String	--

Voraussetzungen:	
Hardware PC	Standard Netzwerkkarte
Software PC	--
Hardware Steuerung	--
Software Steuerung	Benötigt spezielle Software in der B+R System 2000 Steuerung
Benötigt v-dll	--

Plattformen:	
Betriebssysteme	Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server 2012R2;
CE Plattformen	-;

4. Treiber-Historie

Datum	Treiberversion	Änderung
07.07.08	1800	Treiberdokumentation wurde neu erstellt

TREIBERVERSIONIERUNG

Mit zenon 7.10 wurde die Versionierung der Treiber verändert. Ab dieser Version gibt es eine versionsübergreifende Build-Nummer. Das ist die Zahl an der 4. Stelle der Dateiversion.
Zum Beispiel: `7.10.0.4228` bedeutet: Der Treiber ist für Version `7.10`, Service Pack `0` und hat die Build-Nummer `4228`.

Erweiterungen oder Fehlerbehebungen werden zukünftig in einem Build eingebaut und sind dann ab der nächsthöheren Build-Nummer verfügbar.

Beispiel

*Eine Treibererweiterung wurde in Build **4228** implementiert. Der Treiber, den Sie im Einsatz haben, verfügt über die Build-Nummer **8322**. Da die Build-Nummer Ihres Treibers höher ist als die Build-Nummer der Erweiterung, ist die Erweiterung enthalten. Die Versionsnummer des Treiber (die ersten drei Stellen der Dateiversion) spielen dabei keine Rolle. Die Treiber sind versionsunabsabhängig*

5. Voraussetzungen

Dieses Kapitel enthält Informationen zu den Voraussetzungen, die für die Verwendung des Treibers erforderlich sind.

5.1 PC

HARDWARE

Netzwerkkarte

Schnittstelle: Ethernet

Protokoll: B&R TCP

SOFTWARE

Die Treiber Datei BrTcp32.exe in das aktuelle Installationsverzeichnis kopieren (wenn nicht bereits vorhanden) und ins TREIBER_DE.XML File über das Tool DriverInfo.exe eintragen.

Die Konfigurationsdatei (lt. Treibereinstellungen) muss am Ziel-PC in das Projektverzeichnis mit übertragen werden. Hierfür muss diese Datei beim Remote-Transport extra angegeben werden.

5.2 Steuerung

HARDWARE

B&R System 2000

Schnittstelle: Ethernet

Protokoll: B&R TCP

Ethernet Verbindung (TCP/IP). In der Steuerung muss der entsprechende Handling-Funktions-baustein installiert und parametriert sein.

6. Konfiguration

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

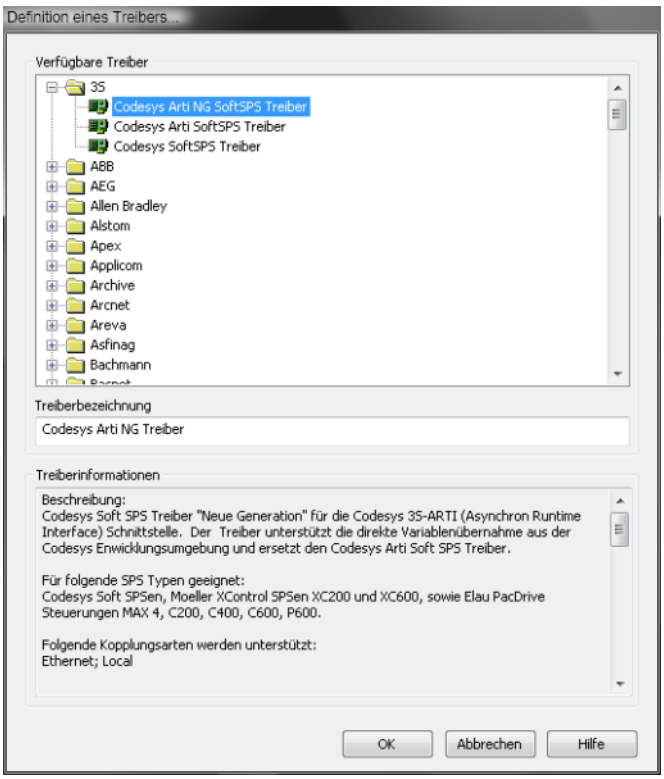
> **Info**
>
> *Weitere Einstellungen, die Sie für Variablen in zenon vornehmen können, finden Sie im Kapitel Variablen (main.chm::/15247.htm) der Online-Hilfe.*

6.1 Anlegen eines Treibers

Um einen neuen Treiber anzulegen:

1. Klicken Sie mit der rechten Maustaste im Projektmanager auf **`Treiber`** und selektieren Sie im Kontextmenü **`Treiber neu`**.

2. In der folgenden Dialogbox bietet Ihnen das Programm eine Auflistung aller verfügbaren Treiber an.



3. 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.
4. 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:

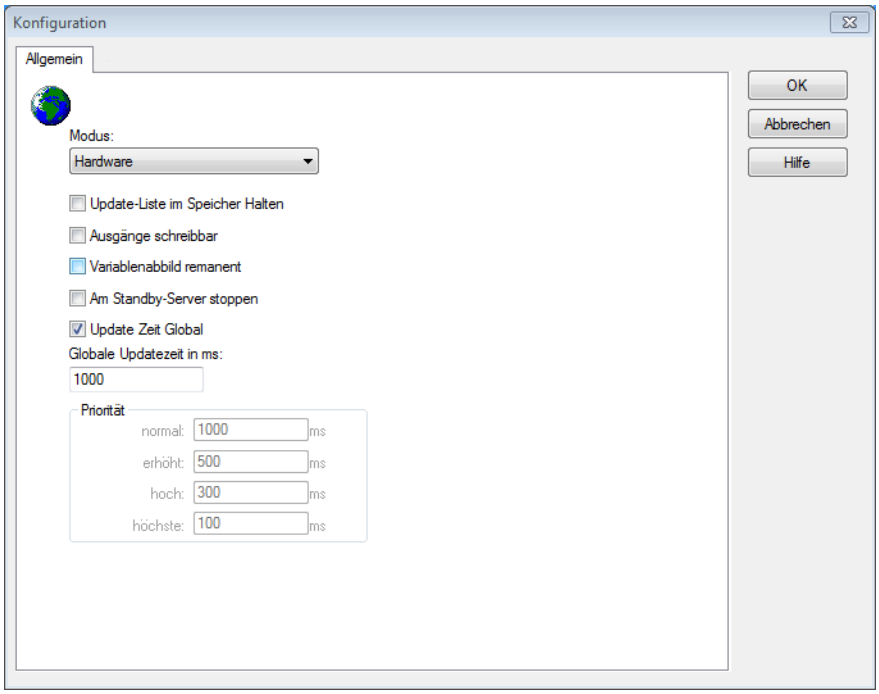
- Intern
- MathDr32
- SysDrv.

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein

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**.



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 oder die Variablen behalten die über zenon Logic gesetzen Werte. Jede Variable hat seinen eigenen Speicherbereich. Zum Beispiel zwei Variablen vom Typ Merker mit Offset 79, können zur Runtime unterschiedliche Werte haben und beeinflussen sich gegenseitig nicht. Ausnahme: Der Simulatortreiber. ▸ Simulation - zählend Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus zählt der Treiber die Werte innerhalb ihres Wertebereichs automatisch hoch. ▸ Simulation - programmiert Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in einer in den Treiber integrierten zenon Logic Runtime ab. Details siehe Kapitel Treibersimulation. (main.chm::/25206.htm)
Update-Liste im Speicher Halten	Einmal angeforderte Variablen werden weiterhin von der Steuerung angefordert, auch wenn diese aktuell nicht mehr benötigt werden. Dies hat den Vorteil, dass z B. mehrmalige Bildumschaltungen nach dem erstmaligen Aufschalten beschleunigt werden, da die Variablen nicht neu angefordert werden müssen. Der Nachteil ist eine erhöhte Belastung der Kommunikation zur Steuerung.
Ausgänge schreibbar	`Aktiv`: Ausgänge können beschrieben werden. `Inaktiv`: Das Beschreiben der Ausgänge wird unterbunden. Hinweis: Steht nicht für jeden Treiber zur Verfügungen.
Variablenabbild remanent	Diese Option speichert und restauriert den aktuellen Wert, den Zeitstempel und die Status eines Datenpunkts. Grundvoraussetzung: Die Variable muss einen gültigen Wert und

	Zeitstempel besitzen. Das Variablenabbild wird im Modus Hardware gespeichert wenn: ▸ einer der Status S_MERKER_1(0) bis S_MERKER8(7), REVISION(9), AUS(20) oder ERSATZWERT(27) aktiv ist Das Variablenabbild wird immer gespeichert wenn: ▸ die Variable vom Objekttyp **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. Hinweis: Nicht verfügbar, wenn CE Terminal als Datenserver dient. Weitere Informationen dazu erhalten Sie im Handbuch zenon Operator im Kapitel CE Terminal als Datenserver.
Update Zeit Global	`Aktiv`: Die eingestellte **Globale Update Zeit** in `ms` wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität wird nicht verwendet. `Inaktiv`: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet.
Priorität	Hier werden die Pollingzeiten der einzelnen Prioritätsklassen eingestellt. Alle Variablen mit der entsprechenden Priorität

	werden in der eingestellten Zeit gepollt. Die Zuordnung zu den Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften. Mit den Prioritätsklassen kann die Kommunikation der einzelnen Variablen auf die Wichtigkeit oder benötigte Aktualität abgestuft werden. Daraus ergibt sich eine verbesserte Verteilung der Kommunikationslast. Achtung: Prioritätsklassen werden nicht von jedem Treiber unterstützt. Zum Beispiel unterstützen spontan kommunizierende Treiber dies nicht.

DIALOG BEENDEN

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

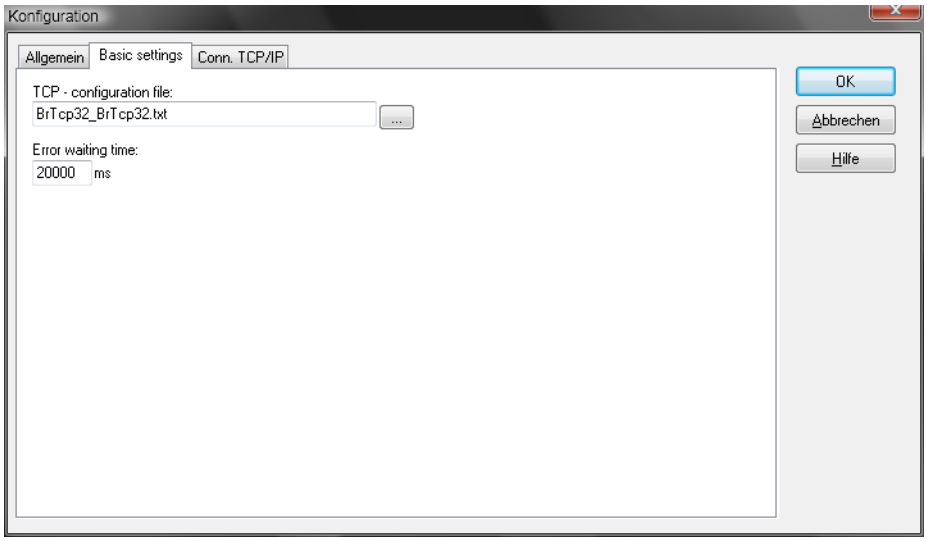
UPDATE ZEIT ZYKLISCHE TREIBER

Für zyklische Treiber gilt:

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

6.2.2 Basic settings

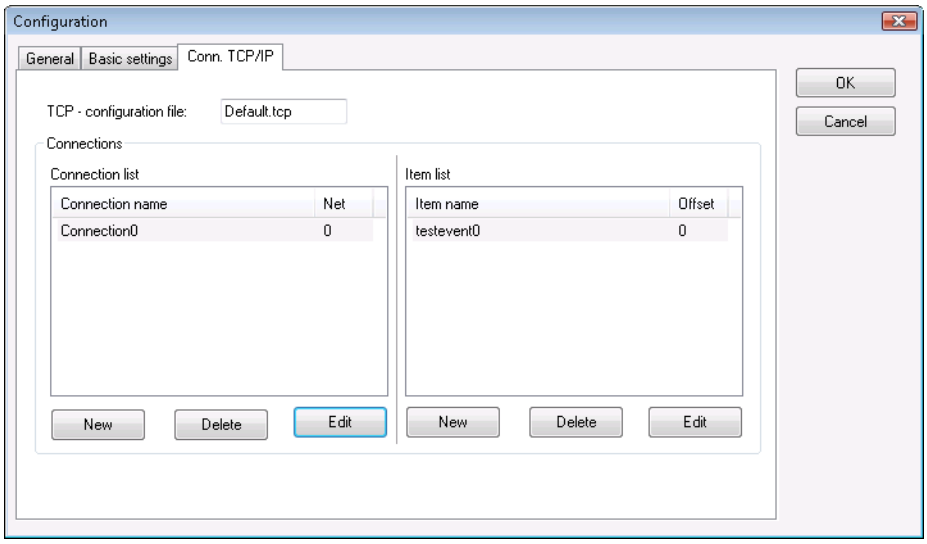
Hier werden allgemeine treiberspezifischen Einstellungen vorgenommen.



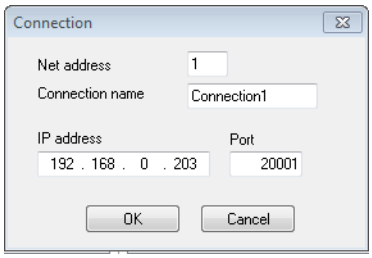
Parameter	Beschreibung
TCP Configuration File	In dieser Datei werden alle Verbindungen und Items gespeichert. Sie muss sowohl am Projektierungsrechner als auch am Laufzeitrechner im Projektverzeichnis vorhanden sein.
Error Waiting Time	Diese Zeit wird bei einem Kommunikationsfehler gewartet. Zudem werden zu jedem Drittel dieser Zeit Lebensmeldungen an die Steuerung gesendet.

6.2.3 Conn. TCP/IP

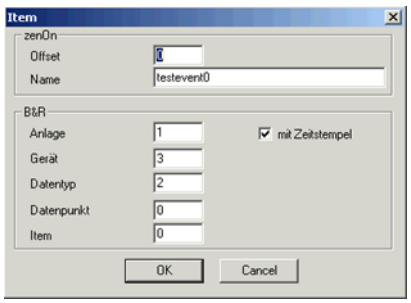
Hier werden alle Verbindungen mitsamt ihrer Items angelegt.



Parameter	Beschreibung
Connection list	In dieser Liste sind alle Verbindungen Projektiert. Wird eine Verbindung selektiert, so er-scheinen in der ‚Item list' die zu ihr projektierten Items.
Item list	In dieser Liste sind alle Items aufgeführt, die zu der aktuell in der ‚connection list' gewählten Verbindung gehören.
Connection	Hier werden die Verbindungseinstellungen parametriert.



Parameter	Beschreibung
Net address	Hier ist eine eindeutige Netzadresse einzugeben, über welche dann eine Variable in zenon dieser Verbindung zugeordnet wird.
Connection name	Frei wählbarer Name.
IP address	IP Adresse der Steuerung.
Port	TCP-Port auf der Steuerung, an dem der Server horcht.



Parameter	Beschreibung
Offset	Hier ist ein eindeutiger Offset einzugeben, über welchen dann eine zenon Variable diesem Item zugeordnet wird.
Name	Frei wählbarer Name.
Anlage	Teil der Adresse in der Steuerung.
Gerät	Teil der Adresse in der Steuerung.
Datentyp	Teil der Adresse in der Steuerung.
Datenpunkt	Teil der Adresse in der Steuerung.
Item	Teil der Adresse in der Steuerung.
Mit Zeitstempel	Item wird mit Zeitstempel angemeldet.

7. Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

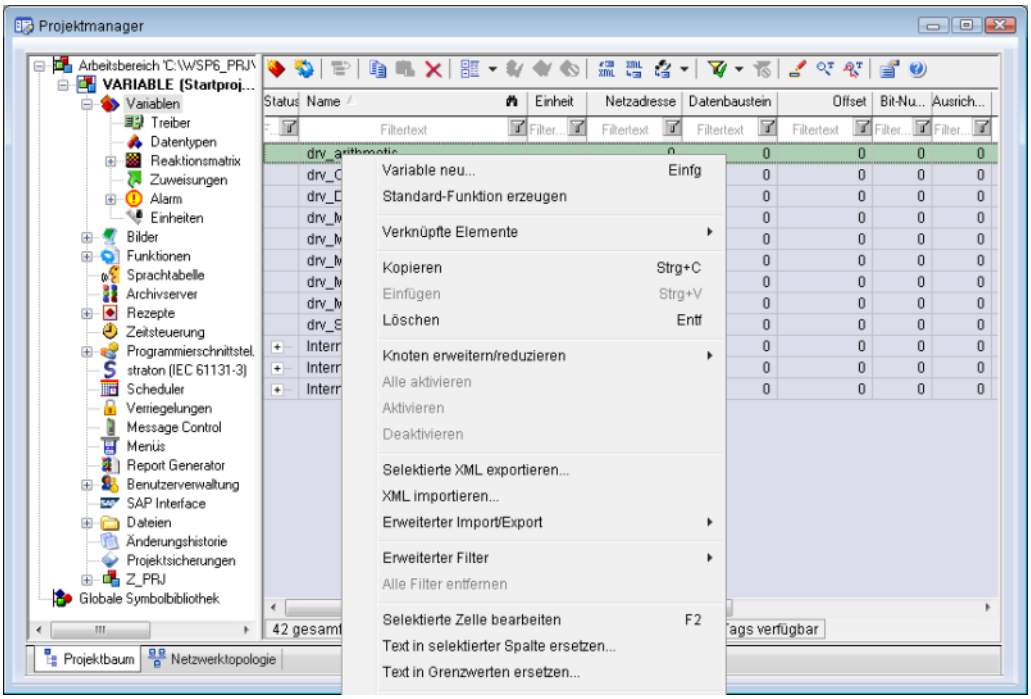
Variablen können angelegt werden:

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

DIALOG VARIABLE

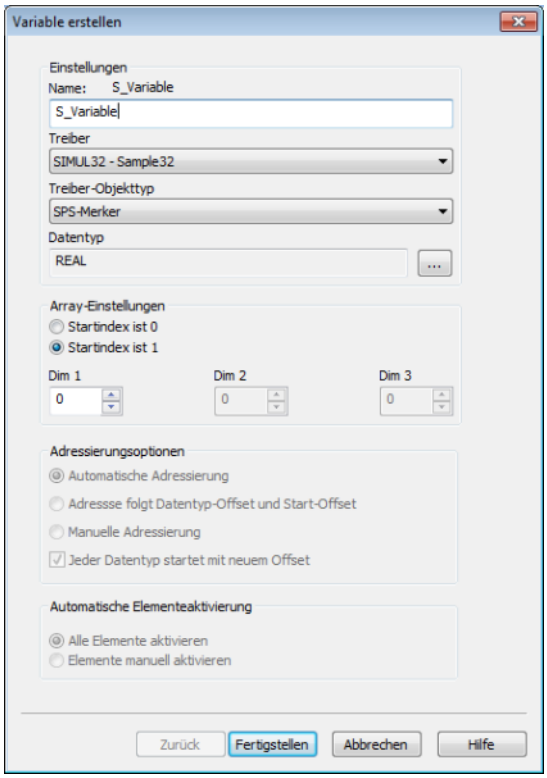
Um eine neue Variable zu erstellen, gleich welchen Typs:

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. 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.
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.

SYMBOLISCHE ADRESSE

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

Maximale Länge: `1024` Zeichen.

ABLEITUNG VOM DATENTYP

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

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

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

7.2 Adressierung

Eigenschaft	Beschreibung
Name	Frei vergebbarer Name; Achtung: je Leitsystemprojekt muss der Name eindeutig sein.
Kennung	Frei vergebbare Kennung, für Betriebsmittelkennung, Kommentar...
Netzadresse	Busadresse oder Netzadresse der Variable. Diese Adresse bezieht sich auf die Busadresse der Verbindungsprojektierung im Treiber. Damit wird ausgewählt auf welcher Steuerung sich die Variable befindet.
Datenbaustein	Für Variablen vom Objekttyp Erweiterter Datenbaustein muss hier die Datenbaustein Nummer angegeben werden. Einstellbar [0.. 4294967295] . Den genauen maximalen Bereich für Datenbausteine sind im Handbuch der Steuerung nachzuschlagen.
Offset	Offset der Variable, die Speicheradresse der Variable in der Steuerung. Einstellbar [0.. 4294967295]
Ausrichtung	wird für diesen Treiber nicht verwendet
Bitnummer	Nummer des Bits innerhalb des eingestellten Offsets. Mögliche Eingabe [0.. 65535], Funktionierender Bereich [0..7]
Stringlänge	Nur verfügbar bei String-Variablen: Maximale Anzahl von Zeichen, die die Variable aufnehmen kann.
Treiber Anbindung/Treiberobjekttyp	Abhängig vom verwendeten Treiber wird bei der Erstellung der Variable ein Objekttyp ausgewählt und kann hier später geändert werden.
Treiber Anbindung/Datentyp	Datentyp der Variable; wird beim Erstellen der Variable ausgewählt und kann hier später verändert werden. ACHTUNG: Wenn der Datentyp nachträglich geändert wird, müssen alle anderen Eigenschaften der Variable überprüft bzw. angepasst werden.
Treiber Anbindung/Priorität	Wird für diesen Treiber nicht verwendet. Der Treiber unterstützt keine zyklisch pollende Kommunikation in Prioritätsklassen.

7.3 Treiberobjekte und Datentypen

Treiberobjekte sind in der Steuerung verfügbare Bereiche wie z. B. Merker, Datenbausteine usw. Hier lesen Sie, welche Treiberobjekte vom Treiber zur Verfügung gestellt werden und welche IEC-Datentypen dem jeweiligen Treiberobjekt zugeordnet werden können.

7.3.1 Treiberobjekte

Folgende Objekttypen stehen in diesem Treiber zur Verfügung:

Treiberobjekttyp	Kanaltyp	Lesen	Schreiben	Unterstützte Datentypen	Kommentar
DATASTREAM-TRIGGER	9	X	X	USINT, SINT	
SETCLOCK-TRIGGER	64	X	X	USINT, SINT	
SPS-Merker	8	X	X	LREAL, BOOL, REAL, DINT, UDINT, INT, UINT, STRING, USINT, SINT	
Treibervariable	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variablen zur statistischen Auswertung der Kommunikation. Weitere Infos finden Sie bei den Treibervariablen (auf Seite 24)

OBJEKTE FÜR PROZESSVARIABLEN IN ZENON

Kanalname	Datenart	Kanaltyp	Objekt	Lesen	Schreiben
BIT	8	8	2	X	X
BYTE	9	8	3	X	X
BYTE _mV	10	8	3	X	X
WORD	2	8	4	X	X
WORD _mV	1	8	4	X	X
DWORD	4	8	5	X	X
DWORD _mV	3	8	5	X	X
FLOAT	5	8	6	X	X
DOUBLE	6	8	6	X	X
STRING	12	8	7	X	X
DATASTREAM-TRIGGER	9	9	8	X	X
SETCLOCK-TRIGGER	9	64	9	--	X

7.3.2 Zuordnung der Datentypen

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

BEISPIELE FÜR ALLE MÖGLICHEN DATENTYPEN IN ZENON

SPS	zenon
I8	i/u8Bit (mit Vorzeichen)
I16	i/u16Bit (mit Vorzeichen)
I32	i/u32Bit (mit Vorzeichen)
U8	i/u8Bit
U16	i/u16Bit
U32	i/u32Bit
F32	float32
F64	double64
Boolean	Boolean
String	String

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

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 Installationsmedium im Ordner `\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 Service Pack-Nummer
RTBuild	UINT	52	zenon Buildnummer
LineStateIdle	BOOL	24.0	TRUE, wenn die Modemleitung belegt ist.
LineStateOffering	BOOL	24.1	TRUE, wenn ein Anruf rein kommt.
LineStateAccepted	BOOL	24.2	Der Anruf wird angenommen.
LineStateDialtone	BOOL	24.3	Rufton wurde erkannt.
LineStateDialing	BOOL	24.4	Wahl aktiv.
LineStateRingBack	BOOL	24.5	Während Verbindungsaufbau.
LineStateBusy	BOOL	24.6	Zielstation besetzt.

LineStateSpecialInfo	BOOL	24.7	Spezielle Statusinformation empfangen.
LineStateConnected	BOOL	24.8	Verbindung hergestellt.
LineStateProceeding	BOOL	24.9	Wahl ausgeführt.
LineStateOnHold	BOOL	24.10	Verbindung in Halten.
LineStateConferenced	BOOL	24.11	Verbindung im Konferenzmodus.
LineStateOnHoldPendConf	BOOL	24.12	Verbindung in Halten für Konferenz.
LineStateOnHoldPendTransfer	BOOL	24.13	Verbindung in Halten für Transfer.
LineStateDisconnected	BOOL	24.14	Verbindung beendet.
LineStateUnknow	BOOL	24.15	Verbindungszustand nicht bekannt.
ModemStatus	UDINT	24	Aktueller Modemstatus.
TreiberStop	BOOL	28	Treiber gestoppt Bei `Treiberstop`, hat die Variable den Wert `TRUE` und ein **OFF**-Bit. Nach dem Treiberstart, hat die Variable den Wert `FALSE` und kein **OFF**-Bit.
SimulRTState	UDINT	60	Informiert über Status der Runtime bei Treibersimulation.

KONFIGURATION

Name aus Import	Typ	Offset	Erklärung
ReconnectInRead	BOOL	27	Wenn TRUE, dann wird beim Lesen automatisch ein Neuaufbau der Verbindung durchgeführt.
ApplyCom	BOOL	36	Änderungen an den Einstellungen der seriellen Schnittstelle zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyCom zur Folge (aktuell ohne weitere Funktion).
ApplyModem	BOOL	37	Änderungen an den Modemeinstellungen zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyModem zur Folge. Diese schließt die aktuelle Verbindung und öffnet

			eine neue entsprechend den Einstellungen **PhoneNumberSet** und **ModemHwAdrSet**.
PhoneNumberSet	STRING	38	Telefonnummer, welche verwendet werden soll.
ModemHwAdrSet	DINT	39	Hardwareadresse, welche zu der Telefonnummer gehört.
GlobalUpdate	UDINT	3	Updatezeit in Millisekunden (ms).
BGlobalUpdaten	BOOL	4	TRUE, wenn die Updatezeit global ist.
TreiberSimul	BOOL	5	TRUE, wenn der Treiber in Simulation ist.
TreiberProzab	BOOL	6	TRUE, wenn das Prozessabbild gehalten werden soll.
ModemActive	BOOL	7	TRUE, wenn das Modem bei diesem Treiber aktiv ist.
Device	STRING	8	Name der seriellen Schnittstelle oder Name des Modem.
ComPort	UINT	9	Nummer der seriellen Schnittstelle.
Baudrate	UDINT	10	Baudrate der seriellen Schnittstelle.
Parity	SINT	11	Parität der seriellen Schnittstelle.
ByteSize	USINT	14	Bitanzahl pro Zeichen der seriellen Schnittstelle. Wert = `0`, wenn der Treiber keine serielle Kommunikation herstellen kann.
StopBit	USINT	13	Anzahl der Stoppbits der seriellen Schnittstelle.
Autoconnect	BOOL	16	TRUE, wenn die Modemverbindung automatisch beim Lesen/Schreiben aufgebaut werden soll.
PhoneNumber	STRING	17	Aktuelle Telefonnummer.
ModemHwAdr	DINT	21	Hardwareadresse zur aktuellen Telefonnummer.

RxIdleTime	UINT	18	Wenn länger als diese Zeit in Sekunden (s) erfolgreich kein Datenverkehr stattfindet, wird die Modemverbindung beendet.
WriteTimeout	UDINT	19	Maximale Schreibdauer bei einer Modemverbindung in Millisekunden (ms).
RingCountSet	UDINT	20	So oft läutet ein hereinkommender Anruf, bevor dieser angenommen wird.
ReCallIdleTime	UINT	53	Wartezeit zwischen Anrufen in Sekunden (s).
ConnectTimeout	UINT	54	Zeit in Sekunden (s) für Verbindungsaufbau.

STATISTIK

Name aus Import	Typ	Offset	Erklärung
MaxWriteTime	UDINT	31	Längste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MinWriteTime	UDINT	32	Kürzeste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MaxBlkReadTime	UDINT	40	Längste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
MinBlkReadTime	UDINT	41	Kürzeste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
WriteErrorCount	UDINT	33	Anzahl der Schreibfehler.

ReadSucceedCount	UDINT	35	Anzahl der erfolgreichen Leseversuche.
MaxCycleTime	UDINT	22	Längste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
MinCycleTime	UDINT	23	Kürzeste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
WriteCount	UDINT	26	Anzahl der Schreibversuche.
ReadErrorCount	UDINT	34	Anzahl der fehlerhaften Leseversuche.
MaxUpdateTimeNormal	UDINT	56	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Normal` in Millisekunden (ms).
MaxUpdateTimeHigher	UDINT	57	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Höher` in Millisekunden (ms).
MaxUpdateTimeHigh	UDINT	58	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Hoch` in Millisekunden (ms).
MaxUpdateTimeHighest	UDINT	59	Zeit seit letzter Aktualisierung der Prioritätsgruppe `Höchste` in Millisekunden (ms).
PokeFinish	BOOL	55	Geht für eine Abfrage auf `1`, wenn alle anstehenden Pokes ausgeführt wurden.

FEHLERMELDUNGEN

Name aus Import	Typ	Offset	Erklärung
ErrorTimeDW	UDINT	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler auftrat.
ErrorTimeS	STRING	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler als String auftrat.
RdErrPrimObj	UDINT	42	Nummer des PrimObjektes, als der letzte Lesefehler verursacht wurde.
RdErrStationsName	STRING	43	Name der Station, als der letzte Lesefehler verursacht wurde.

RdErrBlockCount	UINT	44	Anzahl der zu lesenden Blöcke, als der letzte Lesefehler verursacht wurde.
RdErrHwAdresse	DINT	45	Hardwareadresse, als der letzte Lesefehler verursacht wurde.
RdErrDatablockNo	UDINT	46	Bausteinnummer, als der letzte Lesefehler verursacht wurde.
RdErrMarkerNo	UDINT	47	Merkernummer, als der letzte Lesefehler verursacht wurde.
RdErrSize	UDINT	48	Blockgröße, als der letzte Lesefehler verursacht wurde.
DrvError	USINT	25	Fehlermeldung als Nummer.
DrvErrorMsg	STRING	30	Fehlermeldung als Klartext.
ErrorFile	STRING	15	Name der Fehlerprotokolldatei.

8. Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

INI EINTRÄGE

ZENON6.INI

[BrTcp32]

LOGFILE=1

Aktiviert das spezielle erweiterte Logging des Treibers. Ansicht der Einträge über den DiagnoseViewer (siehe -> Analysetools (auf Seite 34))

EINSCHRÄNKUNG

Wird die Einstellung „Abbild im Speicher halten" deaktiviert, so können Requests nicht korrekt abgearbeitet werden, da in diesem Fall immer der letztgültig von der Steuerung zurückgemeldet wird.

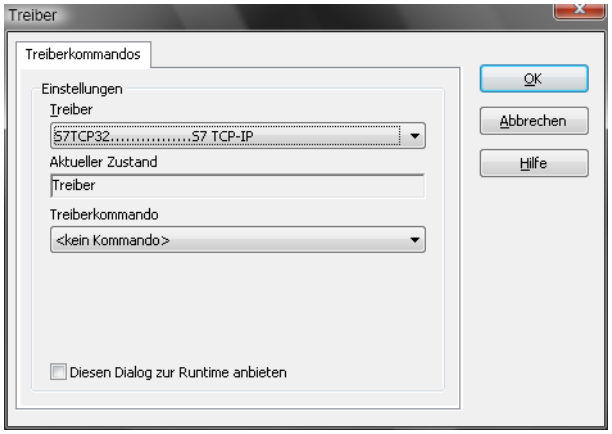
Es muss zumindest immer eine aktive Variable pro Verbindung im Treiber angemeldet sein, ansonsten wird die Verbindung nicht mehr bedient und es werden auch keine Watchdogs etc. mehr versendet.

9. Treiberkommandos

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

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

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



Parameter	Beschreibung
Treiber	Dropdownliste mit allen im Projekt geladenen Treibern.
Aktueller Zustand	Fixer Eintrag, in aktuellen Versionen ohne Funktion.
Treiberkommando	Dropdownliste zur Auswahl des Kommandos.
▸ `Treiber starten (Online-Modus)`	Treiber wird neu initialisiert und gestartet.
▸ `Treiber stoppen (Offline-Modus)`	Treiber wird angehalten, es werden keine neuen Daten angenommen. Hinweis: Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status `Abgeschaltet` (`OFF`; Bit 20).
▸ `Treiber in Simulationsmodus`	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
▸ `Treiber in Hardwaremodus`	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z. B. SPS, Bussystem, ...) angezeigt.
▸ `Treiberspezifisches Kommando`	Eingabe treiberspezifischer Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
▸ `Treiber Sollwertsetzen aktivieren`	Sollwert setzen auf Treiber ist erlaubt.
▸ `Treiber Sollwertsetzen deaktivieren`	Sollwert setzen auf Treiber wird verhindert.
▸ `Verbindung mit Modem aufbauen`	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
▸ `Verbindung mit Modem trennen`	Verbindung beenden (für Modem-Treiber).
Diesen Dialog zur Runtime anbieten	Dialog wird zur Runtime für Änderungen angeboten.

TREIBERKOMMANDOS IM NETZWERK

Wenn sich der Rechner, auf dem die Funktion **`Treiberkommandos`** ausgeführt wird, im zenon Netzwerk befindet, werden zusätzliche Aktionen ausgeführt. Ein spezielles Netzwerkkommando wird vom Rechner zum Server des Projekts gesendet, der dann die gewünschte Aktion auf seinem Treiber durchführt. Zusätzlich sendet der Server das gleiche Treiberkommando zum Standby des Projekts. Der Standby führt die Aktion auch auf seinem Treiber aus.

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

10. Fehleranalyse

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

10.1 Analysetool

Alle zenon Module wie z. B. Editor, Runtime, Treiber, usw. schreiben Meldungen in eine gemeinsame LOG-Datei. Um sie korrekt und übersichtlich anzuzeigen, benutzen Sie das Programm Diagnose Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter Start/Alle Programme/zenon/Tools 7.50 -> 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:

`%ProgramData%\COPA-DATA\LOG`. 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:

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

Hinweise:

1. Der Diagnose Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.
2. 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.
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 `Diagnose Viewers` weiter aufgezeichnet.

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

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

10.2 Fehlernummern

B&R FEHLERCODES

Fehlermeldung	Code	Fehlertext	Beschreibung
BRT_ERR_INVALIDAK	1	Unbekannte Auftragskennung	Der Server hat einen Auftrag mit einer unbekannten Auftragskennung empfangen
BRT_ERR_NO_ITEM	2	Item nicht gefunden	Die Itemadresse in einem Auftrag vom Client ist fehlerhaft (Diese Meldung wird nicht mehr verwendet und wurde durch die Meldungen mit den Fehlercodes 14 bis 18 ersetzt)
BRT_ERR_WRONG_LEN	3	Falsche Datenlänge	Die Längenangabe in einem Auftrag vom Client stimmt nicht mit der Länge des adressierten Items am Server überein
BRT_ERR_WRONG_HYST	4	Falsche Hysterese	Bei der Item-Anmeldung wurde vom Client eine unzulässige Hysterese vorgegeben
BRT_ERR_IS_LOGGED	5	Item ist bereits angemeldet	Der Client versucht ein Item anzumelden, das bereits angemeldet ist
BRT_ERR_ITEM_NO_TS	6	Item hat keinen Zeitstempel	Der Client versucht ein Item mit Zeitstempel anzumelden oder zu lesen, das im Server keinen Zeitstempel besitzt
BRT_ERR_ITEM_NO_LOGON	7	Item ist bereits abgemeldet	Der Client versucht ein Item abzumelden, das bereits abgemeldet ist
BRT_ERR_NO_STREAM	8	Kein Datenstrom	Beim Lesen oder bei der spontanen Übermittlung eines Datenstromes verweist die angegebene Adresse nicht auf einen Datenstrom
BRT_ERR_RTC_SET	9	Schreibfehler von Echtzeituhr	Die Echtzeituhr des Servers konnte nicht gestellt werden

BRT_ERR_ITEM_NOT_WRITEABLE	10	Item schreibgeschützt	Der Server hat einen Schreibauftrag für ein schreibgeschütztes Item erhalten
BRT_ERR_WRITE_PIPE_FULL	11	Schreibpuffer ist voll	Die maximale Anzahl unerledigter Schreibaufträge ist am Server erreicht
BRT_ERR_ITEM_NO_DATA	12	Item hat keine gültigen Daten	Zum Zeitpunkt der Anmeldung durch den Client liegen zu dem adressierten Item noch keine gültigen Daten vor
BRT_ERR_SWC_SET	13	Schreibfehler von SW-Uhr	Die Softwareuhr des Servers konnte nicht gestellt werden
BRT_ERR_UNKNOWN_UNIT	14	Unbekannte Anlagennummer	Der Server hat einen Auftrag mit einer unbekannten Anlagennummer empfangen
BRT_ERR_UNKNOWN_DEVICE	15	Unbekanntes Gerät	Der Server hat einen Auftrag mit einer unbekannten Gerätenummer empfangen
BRT_ERR_UNKNOWN_TYPE	16	Unbekannter Datentyp	Der Server hat einen Auftrag mit einem unbekannten Datentypindex empfangen
BRT_ERR_UNKNOWN_POINT	17	Unbekannter Datenpunkt	Der Server hat einen Auftrag mit einem unbekannten Datenpunktindex empfangen
BRT_ERR_WRONG_ITEM	18	Falsches Item	Der Server hat einen Auftrag mit einem unbekannten bzw. ungültigen Itemindex empfangen

ERROR CODES IN THE API

The following is a list of possible error codes returned by the WSAGetLastError call, along with their ex-tended explanations. Errors are listed in alphabetical order by error macro. Some error codes defined in Winsock2.h are not returned from any function-these are not included in this topic.

Error (Code)	Meaning	Description
WSAEACCES (10013)	Permission denied.	An attempt was made to access a socket in a way forbidden by its access permissions. An example is using a broadcast address for sendto without broadcast permission being set using set-sockopt(SO_BROADCAST). Another possible reason for the WSAEACCES error is that when the bind function is called (on Windows NT 4 SP4 or later), another application, service, or kernel mode driver is bound to the same address with exclusive access. Such exclusive access is a new feature of Windows NT 4 SP4 and later, and is imple-mented by using the SO_EXCLUSIVEADDRUSE option.
WSAEADDRINUSE (10048)	Address already in use.	Typically, only one usage of each socket address (protocol/IP address/port) is permitted. This error oc-curs if an application attempts to bind a socket to an IP address/port that has already been used for an existing socket, or a socket that was not closed properly, or one that is still in the process of closing. For server applications that need to bind multiple sockets to the same port number, consider using set-sockopt(SO_REUSEADDR). Client applications usually need not call bind at all-connect chooses an unused port automatically. When bind is called with a wildcard address (involving ADDR_ANY), a WSAEADDRINUSE error could be delayed until the specific address is committed. This could happen with a call to another function later, including connect, listen, WSAConnect, or WSAJoinLeaf.
WSAEADDRNOTAVAIL (10049)	Cannot assign requested address.	The requested address is not valid in its context. This normally results from an attempt to bind to an address that is not valid for the local machine. This can also result from connect, sendto, WSAConnect, WSAJoinLeaf, or WSASendTo when the remote address or port is not valid for a remote machine (for example, address or port 0).
WSAEAFNOSUPPORT (10047)	Address family not supported by protocol family.	An address incompatible with the requested protocol was used. All sockets are created with an associ-ated address family (that is, AF_INET for Internet Protocols) and a generic protocol type (that is, SOCK_STREAM). This error is returned if an incorrect protocol is explicitly requested in the socket call, or if an address of the wrong family is used for a socket, for example, in sendto.
WSAEALREADY (10037)	Operation already in progress.	An operation was attempted on a nonblocking socket with an operation already in progress-that is, calling connect a second time on a nonblocking socket that is already connecting, or canceling an asynchronous request (WSAAsyncGetXbyY) that has already been canceled or completed.
WSAECONNABORTED	Software caused	An established connection was aborted by the software in your

(10053)	connection abort.	host machine, possibly due to a data transmission time-out or protocol error.

WSAECONNREFUSED (10061)	Connection refused.	No connection could be made because the target machine actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host-that is, one with no server application running.
WSAECONNRESET (10054)	Connection reset by peer.	An existing connection was forcibly closed by the remote host. This normally results if the peer application on the remote host is suddenly stopped, the host is rebooted, or the remote host uses a hard close (see setsockopt for more information on the SO_LINGER option on the remote socket.) This error may also result if a connection was broken due to keep-alive activity detecting a failure while one or more operations are in progress. Operations that were in progress fail with WSAENETRESET. Subsequent operations fail with WSAECONNRESET.
WSAEDESTADDRREQ (10039)	Destination address required.	A required address was omitted from an operation on a socket. For example, this error is returned if sendto is called with the remote address of ADDR_ANY.
WSAEFAULT (10014)	Bad address.	The system detected an invalid pointer address in attempting to use a pointer argument of a call. This error occurs if an application passes an invalid pointer value, or if the length of the buffer is too small. For instance, if the length of an argument, which is a SOCKADDR structure, is smaller than the sizeof(SOCKADDR).
WSAEHOSTDOWN (10064)	Host is down.	A socket operation failed because the destination host is down. A socket operation encountered a dead host. Networking activity on the local host has not been initiated. These conditions are more likely to be indicated by the error WSAETIMEDOUT.
WSAEHOSTUNREACH (10065)	No route to host.	A socket operation was attempted to an unreachable host. See WSAENETUNREACH.
WSAEINPROGRESS (10036)	Operation now in progress.	A blocking operation is currently executing. Windows Sockets only allows a single blocking operation-per- task or thread-to be outstanding, and if any other function call is made (whether or not it references that or any other socket) the function fails with the WSAEINPROGRESS error.
WSAEINTR (10004)	Interrupted function call.	A blocking operation was interrupted by a call to WSACancelBlockingCall.
WSAEINVAL (10022)	Invalid argument.	Some invalid argument was supplied (for example, specifying an invalid level to the setsockopt function). In some instances, it also refers to the current state of the socket-for instance, calling accept on a socket that is not listening.
WSAEISCONN (10056)	Socket is already connected.	A connect request was made on an already-connected socket. Some implementations also return this error if sendto is called on a connected SOCK_DGRAM socket (for SOCK_STREAM sockets, the to pa-rameter in sendto is ignored) although other

		implementations treat this as a legal occurrence.
WSAEMFILE (10024)	Too many open files.	Too many open sockets. Each implementation may have a maximum number of socket handles avail-able, either globally, per process, or per thread.
WSAEMSGSIZE (10040)	Message too long.	A message sent on a datagram socket was larger than the internal message buffer or some other network limit, or the buffer used to receive a datagram was smaller than the datagram itself.
WSAENETDOWN (10050)	Network is down.	A socket operation encountered a dead network. This could indicate a serious failure of the network sys-tem (that is, the protocol stack that the Windows Sockets DLL runs over), the network interface, or the local network itself.
WSAENETRESET (10052)	Network dropped connection on reset.	The connection has been broken due to keep-alive activity detecting a failure while the operation was in progress. It can also be returned by setsockopt if an attempt is made to set SO_KEEPALIVE on a con-nection that has already failed.
WSAENETUNREACH (10051)	Network is unreachable.	A socket operation was attempted to an unreachable network. This usually means the local software knows no route to reach the remote host.
WSAENOBUFS (10055)	No buffer space available.	An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full.
WSAENOPROTOOPT (10042)	Bad protocol option.	An unknown, invalid or unsupported option or level was specified in a getsockopt or setsockopt call.
WSAENOTCONN (10057)	Socket is not connected.	A request to send or receive data was disallowed because the socket is not connected and (when send-ing on a datagram socket using sendto) no address was supplied. Any other type of operation might also return this error-for example, setsockopt setting SO_KEEPALIVE if the connection has been reset.
WSAENOTSOCK (10038)	Socket operation on nonsocket.	An operation was attempted on something that is not a socket. Either the socket handle parameter did not reference a valid socket, or for select, a member of an fd_set was not valid.
WSAEOPNOTSUPP (10045)	Operation not supported.	The attempted operation is not supported for the type of object referenced. Usually this occurs when a socket descriptor to a socket that cannot support this operation is trying to accept a connection on a datagram socket.
WSAEPFNOSUPPORT (10046)	Protocol family not supported.	The protocol family has not been configured into the system or no implementation for it exists. This mes-sage has a slightly different meaning from WSAEAFNOSUPPORT. However, it is interchangeable in most cases, and all Windows Sockets functions that return one of these messages also specify WSAEAFNOSUPPORT.

WSAEPROCLIM **(10067)**	Too many processes.	A Windows Sockets implementation may have a limit on the number of applications that can use it simul-taneously. WSAStartup may fail with this error if the limit has been reached.
WSAEPROTONOSUPPORT **(10043)**	Protocol not supported.	The requested protocol has not been configured into the system, or no implementation for it exists. For example, a socket call requests a SOCK_DGRAM socket, but specifies a stream protocol.
WSAEPROTOTYPE **(10041)**	Protocol wrong type for socket.	A protocol was specified in the socket function call that does not support the semantics of the socket type requested. For example, the ARPA Internet UDP protocol cannot be specified with a socket type of SOCK_STREAM.
WSAESHUTDOWN **(10058)**	Cannot send after socket shutdown.	A request to send or receive data was disallowed because the socket had already been shut down in that direction with a previous shutdown call. By calling shutdown a partial close of a socket is requested, which is a signal that sending or receiving, or both have been discontinued.
WSAESOCKTNOSUPPORT **(10044)**	Socket type not supported.	The support for the specified socket type does not exist in this address family. For example, the optional type SOCK_RAW might be selected in a socket call, and the implementation does not support SOCK_RAW sockets at all.
WSAETIMEDOUT **(10060)**	Connection timed out.	A connection attempt failed because the connected party did not properly respond after a period of time, or the established connection failed because the connected host has failed to respond.
WSATYPE_NOT_FOUND **(10109)**	Class type not found.	The specified class was not found.
WSAEWOULDBLOCK **(10035)**	Resource temporarily unavailable.	This error is returned from operations on nonblocking sockets that cannot be completed immediately, for example recv when no data is queued to be read from the socket. It is a nonfatal error, and the operation should be retried later. It is normal for WSAEWOULDBLOCK to be reported as the result from calling connect on a nonblocking SOCK_STREAM socket, since some time must elapse for the connection to be established.
WSAHOST_NOT_FOUND **(11001)**	Host not found.	No such host is known. The name is not an official host name or alias, or it cannot be found in the data-base(s)

		being queried. This error may also be returned for protocol and service queries, and means that the specified name could not be found in the relevant database.
WSA_INVALID_HANDLE **(OS dependent)**	Specified event object handle is invalid.	An application attempts to use an event object, but the specified handle is not valid.
WSA_INVALID_PARAMETER **(OS dependent)**	One or more parameters are invalid.	An application used a Windows Sockets function which directly maps to a Win32 function. The Win32 function is indicating a problem with one or more parameters.
WSAINVALIDPROCTABLE **(OS dependent)**	Invalid procedure table from service provider.	A service provider returned a bogus procedure table to Ws2_32.dll. (Usually caused by one or more of the function pointers being null.)
WSAINVALIDPROVIDER **(OS dependent)**	Invalid service provider version number.	A service provider returned a version number other than 2.0.
WSA_IO_INCOMPLETE **(OS dependent)**	Overlapped I/O event object not in signaled state.	The application has tried to determine the status of an overlapped operation which is not yet completed. Applications that use WSAGetOverlappedResult (with the fWait flag set to FALSE) in a polling mode to determine when an overlapped operation has completed, get this error code until the operation is com-plete.
WSA_IO_PENDING **(OS dependent)**	Overlapped operations will complete later.	The application has initiated an overlapped operation that cannot be completed immediately. A comple-tion indication will be given later when the operation has been completed.
WSA_NOT_ENOUGH_MEMORY **(OS dependent)**	Insufficient memory available.	An application used a Windows Sockets function that directly maps to a Win32 function. The Win32 func-tion is indicating a lack of required memory resources.
WSANOTINITIALISED **(10093)**	Successful WSAStartup not yet performed.	Either the application has not called WSAStartup or WSAStartup failed. The application may be access-ing a socket that the current active task does not own (that is, trying to share a socket between tasks), or WSACleanup has been called too many times.
WSANO_DATA **(11004)**	Valid name, no data record of requested	The requested name is valid and was found in the database, but it does not have the correct associated data being resolved for. The usual example for this is a

	type.	host name-to-address translation attempt (using gethostbyname or WSAAsyncGetHostByName) which uses the DNS (Domain Name Server). An MX record is returned but no A record-indicating the host itself exists, but is not directly reachable.
WSANO_RECOVERY **(11003)**	This is a nonrecoverable error.	This indicates some sort of nonrecoverable error occurred during a database lookup. This may be be-cause the database files (for example, BSD-compatible HOSTS, SERVICES, or PROTOCOLS files) could not be found, or a DNS request was returned by the server with a severe error.
WSAPROVIDERFAILEDINIT **(OS dependent)**	Unable to initialize a service provider.	Either a service provider's DLL could not be loaded (LoadLibrary failed) or the provider's WSPStartup/NSPStartup function failed.
WSASYSCALLFAILURE **(OS dependent)**	System call failure.	Returned when a system call that should never fail does. For example, if a call to WaitForMultipleObjects fails or one of the registry functions fails trying to manipulate the protocol/name space catalogs.
WSASYSNOTREADY **(10091)**	Network subsystem is unavailable.	This error is returned by WSAStartup if the Windows Sockets implementation cannot function at this time because the underlying system it uses to provide network services is currently unavailable. Users should check: That the appropriate Windows Sockets DLL file is in the current path. That they are not trying to use more than one Windows Sockets implementation simultaneously. If there is more than one Winsock DLL on your system, be sure the first one in the path is appropriate for the network subsystem currently loaded. The Windows Sockets implementation documentation to be sure all necessary components are currently installed and configured correctly.
WSATRY_AGAIN **(11002)**	Nonauthoritative host not found.	This is usually a temporary error during host name resolution and means that the local server did not receive a response from an authoritative server. A retry at some time later may be successful.
WSAVERNOTSUPPORTED	Winsock.dll	The current Windows Sockets implementation does not

(10092)	version out of range.	support the Windows Sockets specification ver-sion requested by the application. Check that no old Windows Sockets DLL files are being accessed.
WSAEDISCON (10101)	Graceful shutdown in progress.	Returned by WSARecv and WSARecvFrom to indicate that the remote party has initiated a graceful shut-down sequence.
WSA_OPERATION_ABORTED (OS dependent)	Overlapped operation aborted.	An overlapped operation was canceled due to the closure of the socket, or the execution of the SIO_FLUSH command in WSAIoctl.

10.3 Checkliste

Ist die Verbindung zur Steuerung korrekt projektiert (IP-Adresse, Port)?

Läuft die Steuerung und ist die Netzwerkverbindung i.O.?

Sind die Items korrekt projektiert?

Stimmen Datentyp der Variablen und des Items in der SPS überein?

Sind Offset und Netz-Adresse in der Variablen korrekt?

Wurde die „Treiberkommunikations-Fehlerdatei" analysiert (Welche Fehler sind aufgetreten) ?

Bei einem Kommunikationsproblem schreibt der Treiber eine genauere Problembeschreibung in eine Treiberkommunikations-Fehlerdatei. Diese Datei wird im Projektverzeichnis (diese befindet sich im Projektpfad unter RT\FILES\zenon\custom\log) angelegt. Der Name dieser Datei _<Treiberbezeichnung>.TXT
Diese Datei kann mit jedem Texteditor z. B. Notepad eingesehen werden.

Zur weiteren Fehleranalyse senden Sie bitte die Projektsicherung und die „Error-Textdatei" an Ihren zuständigen Support.