COPA-DATA
do it your way

zenon Treiber Handbuch

BURPVI

v.8.00

©2018 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. BURPVI 5

3. BURPVI - Datenblatt 6

4. Treiber-Historie 8

5. Voraussetzungen 8

5.1 PC 9

5.2 Steuerung 9

6. Konfiguration 9

6.1 Anlegen eines Treibers 10

6.2 Einstellungen im Treiberdialog 13

6.2.1 Allgemein 14

6.2.2 Global Settings 18

6.2.3 Connections 21

6.2.4 Konfigurationsdateien im Redundanzbetrieb 42

7. Variablen anlegen 43

7.1 Variablen im Editor anlegen 43

7.2 Adressierung 47

7.3 Treiberobjekte und Datentypen 48

7.3.1 Treiberobjekte 49

7.3.2 Zuordnung der Datentypen 50

7.4 Variablen anlegen durch Import 51

7.4.1 XML Import 51

7.4.2 DBF Import/Export 52

7.4.3 PVI-Variablen vom Treiber importieren 59

7.5 Kommunikationsdetails (Treibervariablen) 68

8. Treiberspezifische Funktionen 74

9. Funktion Treiberkommandos 75

10. Fehleranalyse 78

10.1 Analysetool 78

10.2 Checkliste 79

10.3 Fehlermeldungen 80

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. BURPVI

Auf der PVI-Software von Bernecker+Rainer basierender Treiber für Steuerungen von B+R, die mit PVI kommunizieren können. Zum Beispiel System 2000 Familie (2003, 2005 usw.), Acopos, X20 System, Automation PC, PowerPanel usw ...

Der Treiber unterstützt spontanen Betrieb mit Hysterese und Online Import von multidimensionalen Arrays und Array mit einem Startindex <> 0.

Kopplung: Serial, Ethernet

Protokoll: PVI

Der BUR-PVI Treiber ersetzt den BUR20032 Treiber. Auf eine Addressierungskompatibilität zwischen den beiden Treiber wurde nicht geachtet, da der BUR20032 Treiber die PVI Items über eine Zuweisungsdatei adressierte und der BUR-PVI die Adressierung direkt über Eigenschaften in der V-DLL speichert. Der Treiber ist symbolorientiert.

> **Achtung**
>
> *Adressierung eindimensionaler Arrays:*
>
> *Werden eindimensionale Arrays mit einem Startindex <> 0 adressiert, muss an den Index ein Beistrich angehängt werden. Details siehe: Dokumentation B&R.*
>
> ***Zum Beispiel:*** `zenon[3,]`

3. BURPVI - Datenblatt

Allgemein:	
Treiberdateiname	BURPVI.exe
Treiberbezeichnung	BuR-PVI Treiber NG
Steuerungs-Typen	Alle Bernecker und Rainer Steuerungen die mit PVI kommunizieren können, z.B. System 2000 Familie (2003, 2005 usw.), Acopos, X20 System, Automation PC, PowerPanel usw.
Steuerungs-Hersteller	Bernecker + Rainer;

Treiber unterstützt:	
Protokoll	PVI;
Adressierung: Adress-basiert	--
Adressierung: Namens-basiert	X
Kommunikation spontan	X
Kommunikation pollend	X

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

Voraussetzungen:	
Hardware PC	Serielle Schnittstelle RS232 oder Standard Netzwerkkarte
Software PC	PVI Software erforderlich, auch unter Windows CE. Das PC-Setup ist auf dem Installationsmedium verfügbar.
Hardware Steuerung	--
Software Steuerung	--
Benötigt v-dll	X

Plattformen:	
Betriebssysteme	Windows CE 6.0, Embedded Compact 7; Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server 2012R2, Server 2016;
CE Plattformen	x86;

4. Treiber-Historie

Datum	Treiberversion	Änderung
26.01.10	1400	Treiber wurde neu erzeugt
14.01.10	1500	Fehler bei der Konfiguration behoben
05.03.10	1600	Auf PVI 3.0 angepasst
17.03.10	1700	Dialoge überarbeitet

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

PVI-Software von Bernecker+Rainer.

5.2 Steuerung

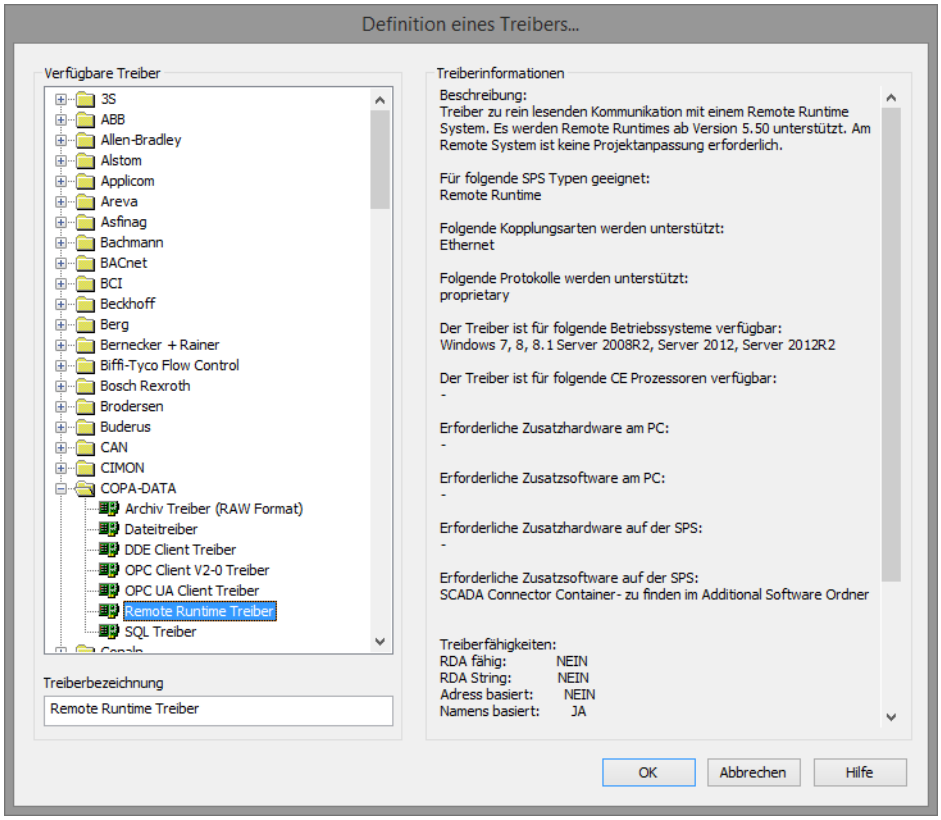
6. Konfiguration

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

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

6.1 Anlegen eines Treibers

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



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

DIALOG BEENDEN

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

Info

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

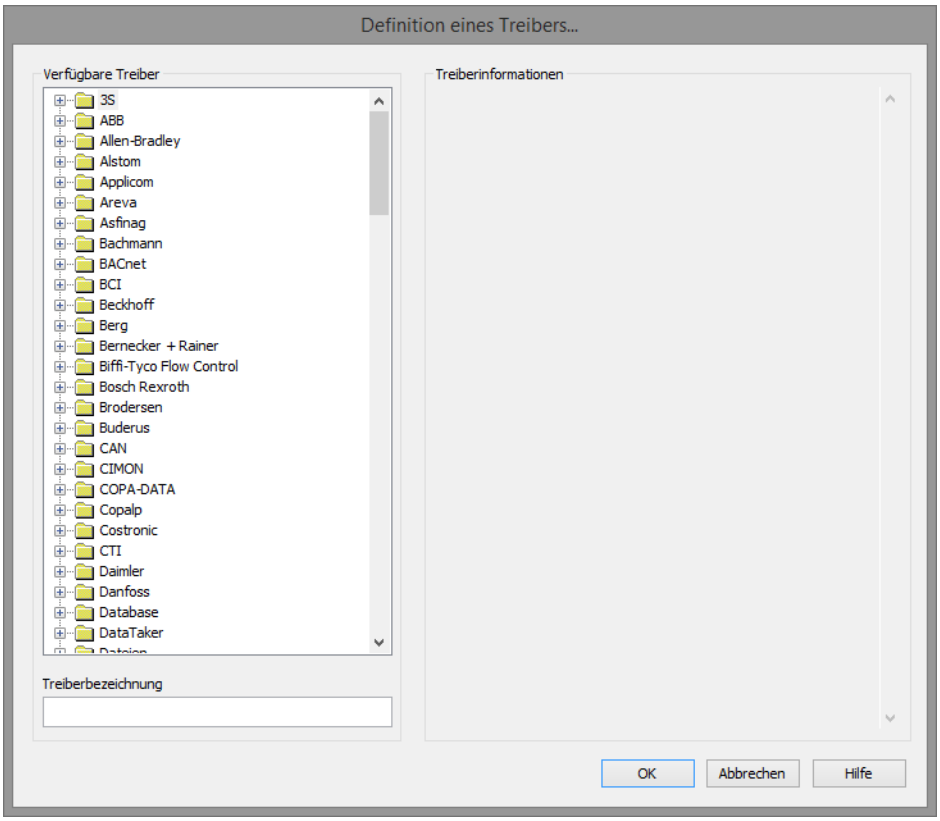
TREIBER NEU ANLEGEN

Um einen neuen Treiber anzulegen:

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

Variablen.
Der Dialog **Treiber erstellen** wird geöffnet.

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



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

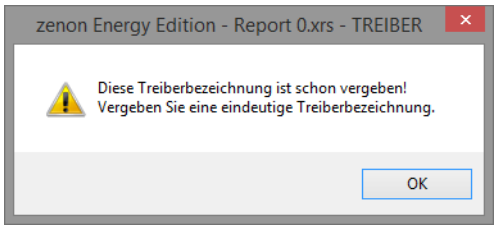
   - Die **Treiberbezeichnung** muss eindeutig sein.
     Wird ein Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
     Dies wird durch Klick auf die Schaltfläche **OK** evaluiert. Ist die Treiber im Projekt bereits vorhanden wird dies mit einem Warndialog angezeigt.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

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

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

DIALOG TREIBERBEZEICHNUNG BEREITS VORHANDEN

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



ZENON PROJEKT

Bei neu angelegten Projekten werden die folgenden Treiber automatisch angelegt:

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

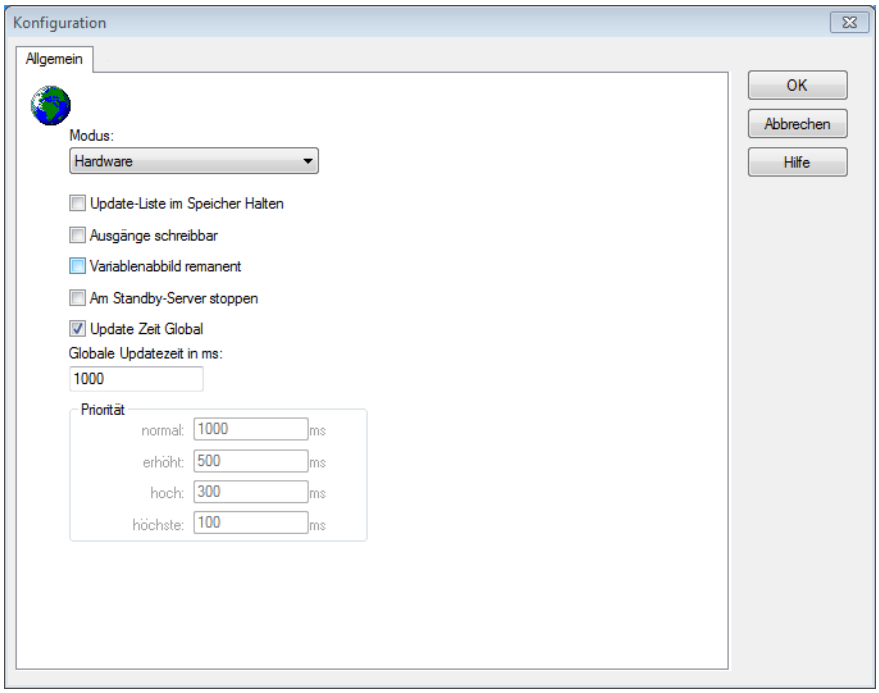
> **Info**
>
> In einem zenon Projekt müssen nur die benötigten Treiber vorhanden sein. Treiber können bei Bedarf zu einem späteren Zeitpunkt hinzugefügt werden.

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein

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



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

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

DIALOG BEENDEN

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

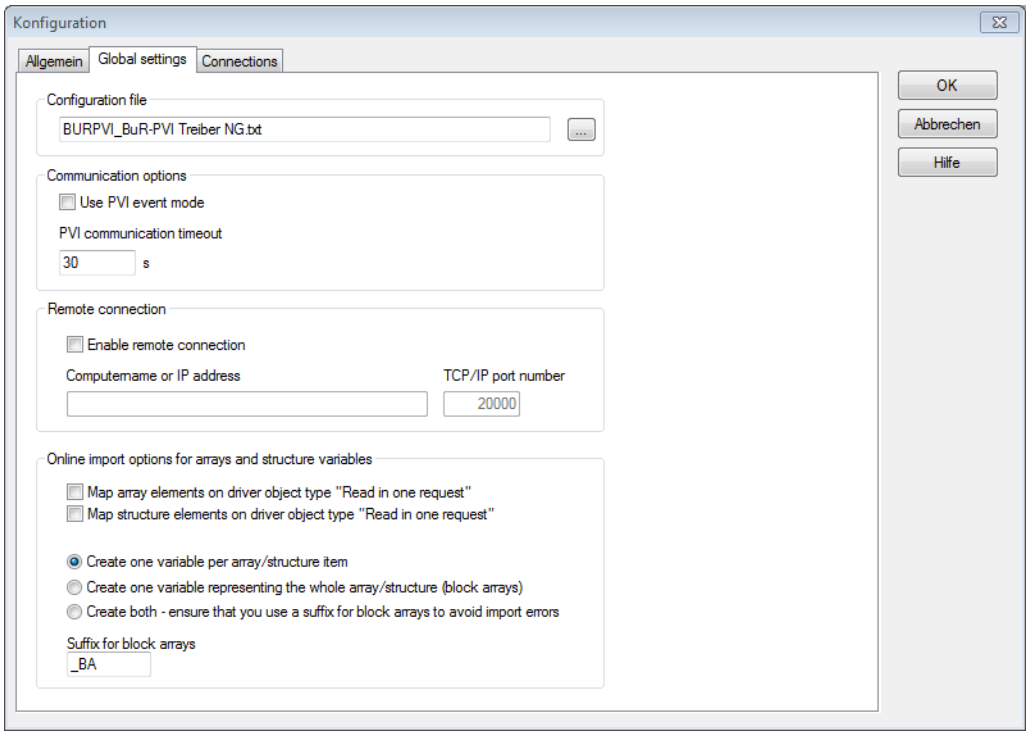
UPDATE ZEIT SPONTANE TREIBER

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

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

6.2.2 Global Settings

In der Registerkarte **Global Settings** werden allgemeine Einstellungen für die PVI-Kommunikation gesetzt. Diese Einstellungen gelten für alle Verbindungen, die im Treiber angelegt sind.



Parameter	Beschreibung
Configuration file	Datei in der die Konfiguration der Verbindungen und die globalen Einstellungen gespeichert werden. Klick auf **...** öffnet den Windows Explorer. Beachten Sie auch die Hinweise im Kapitel: Konfigurationsdateien im Redundanzbetrieb (auf Seite 42).

REMOTE CONNECTION

Parameter	Beschreibung
Use PVI event mode	`Aktiv`: Alle PVI Variablen werden per Eventmodus angefordert (PVI Kommando `AT=rwe HY=0`). `Inaktiv`: PVI Variablen werden mit der Einstellung der globalen Updatezeit (**AT=rw**, **RF**=**Globale Updatezeit in ms**) zyklisch gelesen.
PVI communication timeout [s]	Timeout für PVI Kommunikation in Sekunden. PviInitialize (dwTimeout, …)
Enable Remote Connection	`Aktiv`: Verbindung zu einem Remote PVI System über TCP/IP wird hergestellt. Die Kommunikation erfolgt dann ausschließlich über dieses Gateway. Es ist immer auch eine lokale PVI Installation erforderlich. Zusätzlich zu den Einstellungen am Treiber muss auch beim PVI selbst (PVI Client/Manager) die Einstellung zur Remote Kommunikation über TCP/IP konfiguriert werden (lokal und am Remote-System).
Computername or IP address	Rechnername oder IP Adresse auf dem das Remote PVI System installiert ist und läuft.
TCP/IP Port number	TCP/IP Portnummer des Remote PVI Systems. Hinweis: Eine eventuell vorhandene Firewall muss so konfiguriert sein, dass der Zielport erreichbar ist.

ONLINE IMPORT OPTIONS FOR ARRAYS AND STRUCTURE VARIABLES

Optionen für den Online-Import von Array-Variablen und Strukturvariablen.

Parameter	Beschreibung
Map array elements on driver object Type "Array as whole"	`Aktiv`: Beim Import von Array-Elementen wird als **Treiberobjekttyp** `In einer Abfrage lesen` gewählt - ausgenommen für Arrays in Strukturen.
Map structure elements on driver object Type "Array as whole"	`Aktiv`: Beim Import von Struktur-Elementen wird als **Treiberobjekttyp** `In einer Abfrage lesen` gewählt - auch für Arrays in Strukturen.
Create one variable per array/structure item	`Aktiv`: Es werden nur einzelne Variablen erstellt und keine Knoten dargestellt. Default-Einstellung.
Create one variable representing the whole array/structure (block arrays)	`Aktiv`: Für Arrays und Strukturen werden nur Blockvariablen erstellt, aber keine Variablen, die einzelne Elemente von Strukturen/Arrays repräsentieren. Auf Top-Level Variablen hat diese Option keinen Einfluss.
Create both - ensure that you use a suffix for block arrays to avoid import errors	`Aktiv`: Für Arrays und Strukturen werden sowohl einzelne Elemente als Blocks, die das ganze Array oder die gesamte Struktur repräsentieren, erstellt.
Suffix for block arrays	Suffix für den zenon Namen von Variablen, die als Block-Arrays erzeugt wurden, d.h. Variablen die eine ganze Struktur oder ein komplettes Array repräsentieren.

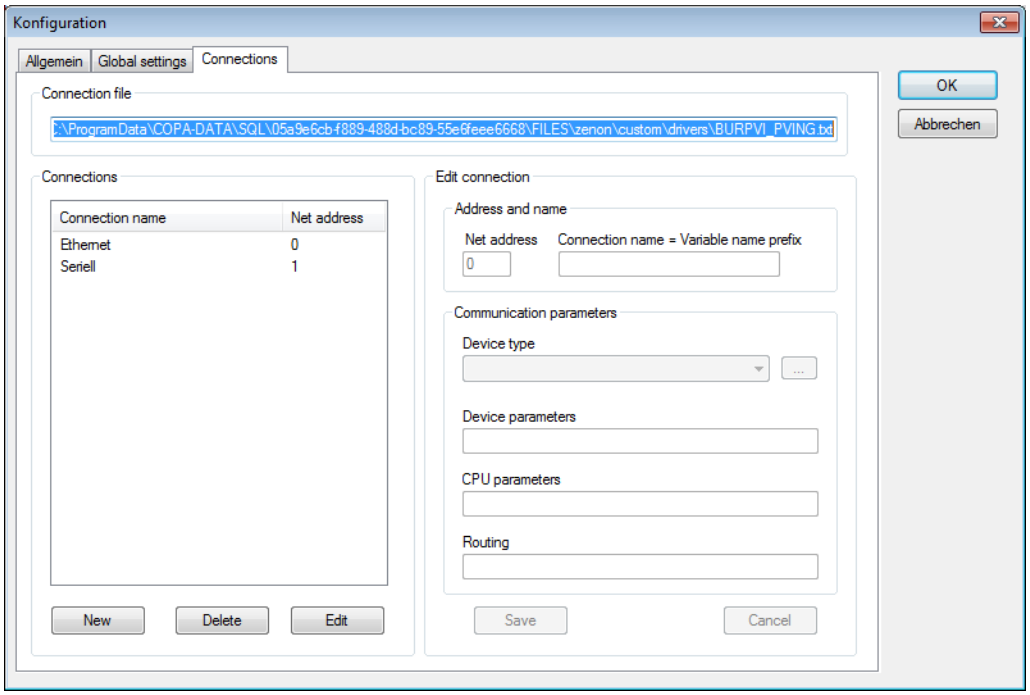
ARRAY AS A WHOLE

Damit werden alle Elemente auf den Treiber Objekttyp **Array als Ganzes** gemappt . Alle Elemente dieses Objekttyps werden nicht mehr bei Wertänderung gelesen, sondern zyklisch gepollt. Der Treiber liest immer den gesamten Array als Block ein.

Ein Block-Array ist eine große Variable in zenon, die einen kompletten Block von der SPS ausliest. Damit werden viele einzelne Array-Variablen als ein einziger Array gelesen.

6.2.3 Connections

In dieser Registerkarte erfolgen die Einstellungen für die Verbindungen. Im Treiber können beliebig viele Verbindungen angelegt werden. Alle Verbindungen nutzen die Globalen Einstellungen, die auf der Registerkarte **General Settings** (auf Seite 18) eingestellt sind.



Parameter	Beschreibung
Connection file	Speicherort für die Konfigurationsdatei in der die Verbindungen und die globalen Einstellungen gespeichert werden. Die Datei wird in der Registerkarte **General Settings** (auf Seite 18) eingestellt.
Connections	Liste der zur Verfügung stehenden Verbindungen.
New	Fügt neuen Eintrag zur Liste hinzu. Einstellungen werden im Feld rechts der Liste konfiguriert.
Delete	Löscht ausgewählten Eintrag aus Liste.
Edit	Ermöglicht Konfiguration des ausgewählten Eintrags. Der Bereich rechts der Liste zum Editieren der Verbindung wird aktiviert. Achtung! Im Editiermodus kann der Treiberdialog nicht geschlossen werden. Erst wenn der Editiermodus mit **Save** oder **Cancel** verlassen wurde ist ein Schließen des Treiberdialogs möglich.
Edit connection	Verbindungseinstellungen editieren.
Adress and name	Adresse und Name.
Net address	Netzadresse der Verbindung für die Zuordnung zur Variablen. Achtung! Wenn Sie die Netzadresse hier ändern, muss auch die Netzadresse bei allen Variablen der Verbindung geändert werden. Sonst stimmt die Zuordnung nicht mehr und die Variablen können in der Runtime nicht kommunizieren!
Connection name	Symbolischer Verbindungsname. Dieser Name wird als Präfix für den Variablennamen verwendet.
Communication parameters	Kommunikationsparameter.
Device type	Auswahl der PVI Gerätetyps aus der Dropdownliste: ▸ Seriell ▸ Ethernet ▸ Modem ▸ CAN ▸ Shared ▸ Userdefined Für die Typen **Seriell** und **Ethernet** kann mittels Klick auf **...** ein Dialog zur Konfiguration geöffnet werden.
Device parameters	Direkteingabe der PVI Device Parameter. Wenn die Einstellungen mit dem Konfigurationsdialog für Ethernet oder Serielle Verbindungen gemacht wird, werden die Einstellungen hier überschrieben. Ethernet Default: `/IF=TCPIP /SA=1`

	Seriell Default: `/IF=COM1 /BD=57600 /PA=2 /IT=1` Die Einstellungen für die einzelnen Kommunikationsarten finden Sie im Kapitel Device Parameter - INA2000 Geräteobjekt (auf Seite 32).
CPU parameters	Direkteingabe der PVI CPU Parameter. Das ist die Stationsadresse der CPU. Für serielle Verbindungen ist keine Stationsadresse erforderlich. Ethernet Deafult: `/DA=2` Seriell Default: -
Routing	Direkteingabe eines Routingpfades für Routing über SPS Stationen. Siehe Beschreibung für Routing weiter unten.
Save	Sichert Konfiguration der ausgewählten Verbindung.
Cancel	Verwirft Konfiguration.
OK	Speichert alle Änderungen und schließt Dialog.
Abbrechen	Verwirft alle Änderungen und schließt Dialog.

Info

Maximale Anzahl der Ethernet-Verbindungen: 256.

ROUTING

Mit Routing können über eine SPS Station Kommunikationsverbindungen zu SPS Stationen anderer Netzwerke hergestellt werden. Diese Verbindungen können auch über mehrere SPS Stationen laufen. Die einzelnen Stationen werden im Routing-Pfad definiert.

SYNTAX

`<Eintrag 1>/<Eintrag 2>/.../<Eintrag n>`
Die einzelnen Pfadeinträge sind durch das Zeichen '/' getrennt.

Syntax für einen Eintrag:

`SL<Slot>.SS<Subslot>.IF<Interface>.<Address>`

Alle Teileinträge sind durch das Zeichen '.' getrennt. Die Angaben SL<Slot> und SS<Subslot> brauchen für 0 bzw 1 nicht angegeben werden. Die Angabe <Address> wird nur für Verbindungen über ein Netzwerk benötigt.

Parameter	Beschreibung
`SL<Slot>`	Slotnummer: 0 (1) - 15.
`SS<Subslot>`	Subslotnummer: 1 - 3.
`IF<Interface>`	Interfacenummer: 1 - 15.
`<Address>`	Stationsadresse: 1 - ff (Hexadezimal).

Routen auf bzw. über Peripherie Prozessoren:

Parameter	Beschreibung
CP	Haupt Prozessor.
PP<Slot>	Peripherie Prozessor, Slotnummer: 0 (1) - 15.

BEISPIELE

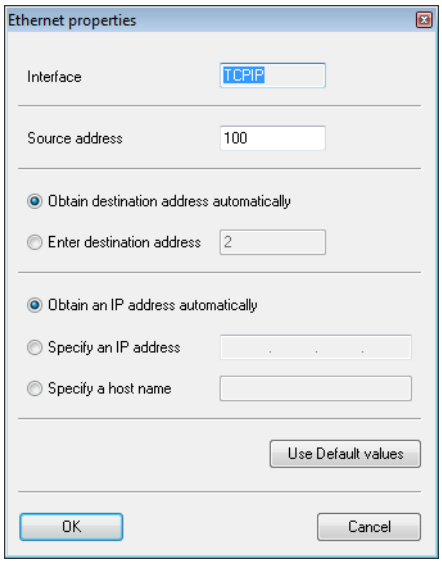
/CN=IF3/IF2.7
Vom PC auf die erste SPS, dort über IF3 weiter zur nächsten Station (SPS) und dann über IF2 auf eine CAN Station mit der Stationsnummer 7.

/CN=SL6.IF1
Vom PC auf die erste SPS, über Steckplatz 6 (z. B. IF 050) und IF1 weiter.

/CN=PP3
Vom PC die auf erste SPS und weiter auf Peripherie Prozessor 3

ETHERNET-VERBINDUNG

Klick auf **...** im Abschnitt **Device type** öffnet bei ausgewähltem **Ethernet** den Dialog zur Konfiguration der Ethernet-Verbindung:

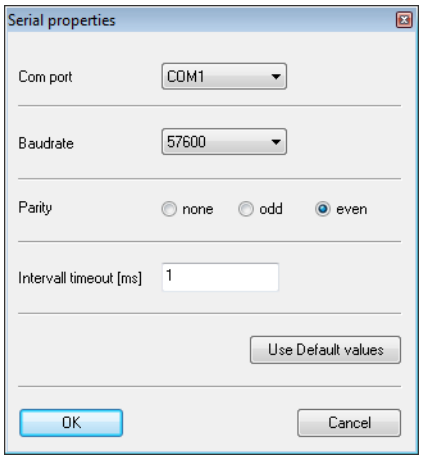


Parameter	Beschreibung
Interface	Definiert die Schnittstelle. Muss immer TCPIP sein.
Source address	Stationsnummer der Quellstation (eigene Station). Achtung! Die Stationsnummer muss innerhalb des INA2000 Netzwerks eindeutig sein. Das B&R Automation Studio verwendet Standardmäßig die Stationsnummer 1. Es kann also sehr leich passieren, dass bei Verwendung der Stationsnummer 1 die Kommunikation zur Steuerung nicht funktioniert. Die höchste Stationsnummer ist 99 bei SG3 und 255 bei SG4. Siehe auch Konfigurationsdateien im Redundanzbetrieb (auf Seite 42).
Obtain destination address automatically	Zieladresse automatisch beziehen: PVI verbindet sich zum eingestellten Gerät (IP Adresse oder Rechnername muss gesetzt sein) und ermittelt die Stationsadresse automatisch.
Enter destination address	Zieladresse eingeben: Geben Sie hier die Stationsadresse des Zielsystems ein. Wenn **Obtain an IP address automatically** gesetzt ist, wird ein Broadcast im Netzwerk durchgeführt, um die Station zu finden.
Obtain an IP address automatically	IP Adresse automatisch beziehen: Führt einen Broadcast im Netzwerk durch und versucht, die Station mit der bei **Enter destination address** eingestellten Stationsadresse zu finden.
Specify an IP address	IP Adresse festlegen: Geben Sie die IP Adresse der Station an.
Specify a host name	Hostnamen festlegen: Geben Sie den Hostnamen der Station an.
Use Default values	Setzt die Eigenschaften auf die Standardeinstellungen zurück.
OK	Übernimmt Einstellungen und schließt Dialog.
Cancel	Verwirft Einstellungen und schließt Dialog.

Alternativ können die Werte auch direkt in die Abschnitte **Device Parameter**, **CPU Parameter** und **Path target system** eingetragen werden.

SERIELLE VERBINDUNG

Klick auf **...** im Abschnitt **Device type** öffnet bei ausgewähltem **Seriell** den Dialog zur Konfiguration der seriellen Verbindung:



Parameter	Beschreibung
Com port	Definiert Schnittstelle. Default: `COM1`
Baudrate	Definiert Baudrate. Default: `57600`
Parity	Definiert Parity: ▸ `no` ▸ `odd` ▸ `even` Default: `even`
Intervall-timeout(ms)	Legt den Intervall in Millisekunden fest. Default: `1`
Use Default values	Setzt die Eigenschaften auf die Standardeinstellungen zurück.
OK	Übernimmt Einstellungen und schließt Dialog.
Cancel	Verwirft Einstellungen und schließt Dialog.

Alternativ können die Werte auch direkt in die Abschnitte **Device Parameter**, **CPU Parameter** und **Path target system** eingetragen werden.

INA2000 Kommunikation

Der Treiber untersützt die Kommunikation nach dem INA2000 Protokoll.
Die INA2000 Kommunikation wird auf folgenden Geräten unterstützt:

- Serielle Kommunikation
- CAN Kommunikation
- Ethernet UDP Kommunikation
- Modem Kommunikation
- Profibus FDL Kommunikation
- LS251 Kommunikation
- AR000 Kommunikation

Jede INA2000 Kommunikationsverbindung besitzt eine Lebensüberwachung. Sobald eine Verbindung hergestellt ist, wird diese jede Sekunde durch Austausch von kleinen Datenpaketen (ping / pong), geprüft.

ALLGEMEINE SPS VORAUSSETZUNGEN:

Die INA2000 Kommunikation funktioniert eingeschränkt (Dienste) ab SPS Betriebssystem 1.91. Um alle Dienste (z. B. Upload, Download von Modulen) verwenden zu können, wird ein SPS Betriebssystem 2.10 benötigt.

Sind Multimaster-Fähigkeit oder eine TCP/IP Kommunikation zwischen PC und SPS gefordert, muss ein SPS Betriebssystem größer/gleich 2.20 verwendet werden. Für eine INA2000 Eventbehandlung über die SPS (SPS überwacht Datenänderung) wird ein SPS Betriebssystem 2.24 oder größer benötigt.

INA2000 Aufträge werden auf der SPS in der Restzeit abgearbeitet. Um geeignete Antwortzeiten zu erhalten, ist mindestens eine Restzeit von 20% Voraussetzung. Können aus applikationstechnischen Gründen keine 20% Restzeit garantiert werden, kommt es zu größeren Antwortzeiten und damit zu einer langsameren Aktualisierung (Refresh) der Daten.

SERIELLE KOMMUNIKATION

Die serielle Kommunikation kann nur als Punkt zu Punkt Verbindung (RS232 oder RS422) betrieben werden.

CAN KOMMUNIKATION

Die CAN Kommunikation wird als INA2000 Netzwerk mit maximal 32 (erweiterbar auf 255) Stationen betrieben. Die einzelnen Stationen werden durch die CAN Knotennummer unterschieden.

PC VORAUSSETZUNGEN

Welche CAN Karten unter welcher Windows-Plattform betrieben werden können, ist der Beschreibung des Herstellers zu entnehmen.

SPS VORAUSSETZUNGEN

Um eine INA2000 Kommunikation über CAN zu erlauben, muss im Automation Studio die CAN Kommunikation aktiviert werden. Siehe Automation Studio Dokumentation (*Projekte -> Hardwarekonfiguration -> CAN Konfiguration*).

ETHERNET UDP KOMMUNIKATION

PC VORAUSSETZUNGEN

Es muss das TCP/IP Protokoll installiert sein.

SPS VORAUSSETZUNGEN

Um eine INA2000 Kommunikation über Ethernet zu erlauben, muss im Automation Studio die Ethernet Kommunikation aktiviert werden. Siehe Automation Studio Dokumentation (Projekte -> Hardwarekonfiguration -> Ethernet Konfiguration).

Für SG4 (i386) gilt:
Die Konfiguration der IP Adresse und der Stationsnummer (INA Knotennummer) erfolgt im Ethernet Eigenschaftsdialog des Automation Studios. Es müssen keine zusätzlichen Module auf die SPS übertragen werden.

Für SG3 (m68k) gilt:
Für die INA2000 Kommunikation müssen die Module "FBTCPIP.BR" und "TCPIPMGR.BR" gebrannt werden. Zusätzlich muss ein Datenobjekt "TCPIPCFG.DAT" mit den entsprechenden Einstellungen existieren.

MEHRERE ETHERNET NETZWERKKARTEN AM PC VERWENDEN

Wie kann man sicherstellen, dass PVI die richtige Netzwerkkarte verwendet um auf die Steuerung zu kommunizieren, wenn zwei oder mehr Netzwerkkarten im PC stecken.

Damit die richtige Netzwerkkarte für die SPS Kommunikation gewählt wird muss die Subnet Maske richtig eingestellt werden. Die IP Adresse der Zielstation (SPS) wird mit der Subnet Maske mit UND verknüpft, dadurch wird die richtige Netzwerkkarte gefunden und für die Kommunikation verwendet.

Beispiel:

Device Parameter: /IF=tcpip /SA=1

CPU Parameter /DA=13 /DAIP=172.43.70.13

IPAdr NW1: 172.43.71.12 SubNet Mask: 255.255.255.0

IPAdr NW2: 172.43.70.12 SubNet Mask: 255.255.255.0

IPAdr SPS: 172.43.70.13

Damit wird zur Kommunikation zur SPS die Netzwerkkarte zwei herangezogen. Wird für eine Netzwerkkarte ein DHCP Server konfiguriert, so muss bekannt sein, welche IP Adressen und SubNet Masken der DHCP Server vergibt, damit man die zweite Netzwerkkarte richtig konfigurieren kann.

MODEM KOMMUNIKATION

Die Modem Kommunikation ist wie die serielle Kommunikation auch nur eine Punkt zu Punkt Verbindung.

Modem Features:

- Transparenz: Auf das Modem wird nur über Standard Windows-Funktionen zugegriffen. D.h. es kann jedes Modem verwendet werden, dass unter Windows installiert werden kann. Der Anwender braucht sich nur um die Installation des Modems zu kümmern und benötigt ansonsten keine Kenntnisse bezüglich Modems (AT-Befehlssatz, etc.).
- Automatische Verbindungswiederherstellung: Eine aktiv erstellte Verbindung (s.u.) wird laufend überwacht. Bei Verlust der Verbindung (z. B. Unterbrechung der Telefonleitung, Abstecken des Modems, Ausschalten des Modems) wird in periodischen Abständen versucht, die Verbindung wieder aufzubauen. Die Anzahl der Versuche sowie der zeitliche Abstand dazwischen ist parametrierbar.
- Aktiver Verbindungsaufbau: Der Windows Rechner stellt zu einer angegebenen Telefonnummer eine Wählverbindung her und überträgt auf dieser Verbindung das durch die PVI Linie festgelegte Protokoll.
- Passiver Verbindungsaufbau: Der Windows Rechner wartet auf ankommenden Ruf, hebt selbsttätig ab und baut die Verbindung auf. Mit diesem Betriebsmodus ist es z. B. möglich, dass eine SPS den Verbindungsaufbau zu einem Windows Rechner mit PVI selbst initiiert.

PC VORAUSSETZUNGEN

Das PVI Modem-Gerät benötigt Microsoft TAPI Version 2.0. Diese Version ist bei Windows NT 4.0 bereits installiert. Beim Betrieb unter Windows 95 ist es notwendig, die standardmäßig mitinstallierte TAPI Version 1.4 durch eine neuere Version zu ersetzen. Die aktuell von Microsoft (www.microsoft.com) verfügbare Version trägt die Versionsnummer 2.1. Um TAPI Version 2.1 unter Windows 95 zu installieren, ist das Programm " tapi2195.exe" im Unterverzeichnis "Pvi\SysSetup\Modem" zu starten. Diese Datei wird mit der PVI Setup-Option "Modem Systemkomponenten" installiert.

Die Installation eines Modems geschieht in der Windows Systemsteuerung -> Modem. Im geöffneten Dialog ist es möglich, neue Modems hinzuzufügen und die Einstellung (Punkt "Eigenschaften") des Modems zu ändern. Die Bezeichnung des Modems in diesem Dialog entspricht dem Modemnamen, der als PVI Geräteparameter (/MO) angegeben werden muss.
Das Verhalten beim Wählen kann unter "Systemsteuerung -> Telefon" konfiguriert werden (aktueller Standort,Wahlkarte, etc.).

PROFIBUS FDL KOMMUNIKATION

Achtung:

Die INA2000 Kommunikation über Profibus FDL ist nur auf dem System 2000 möglich. Fremdsysteme werden von PVI nicht unterstützt !

Die Einstellungen der SPS Profibus Karten (NW100/NW150) und der PC Karte (5A1104.00-090) sind dem PROFIBUS Anwenderhandbuch (MASYS2PB-0) zu entnehmen.

PC VORAUSSETZUNGEN

Der Datenaustausch mit dem PC erfolgt über ein 32 kByte großes DPR. Hierfür muss ein freier Adressbereich auf dem PC eingestellt werden. Auf der Profibuskarte wird dies mit den zwei Hex-Schaltern SW1 und SW2 eingestellt (SW1 = 0; SW2=D). Wird nur eine Profibus Karte im System betrieben, sind die Jumper BR1 und BR2 offen zu lassen. Weitere Informationen sind dem PROFIBUS Anwenderhandbuch zu entnehmen.

Damit dieser Speicher auch auf dem PC freigegeben ist, muss im BIOS dieser Speicherbereich excludiert werden. Auf dem IPC5000 müssen im BIOS unter "PNP/PCI Configuration" entweder 32 kByte (1 Profibus Karte) oder 64 kByte (bei 2 Profibus Karten) ab der Basis Adresse D000 excludiert werden.

SPS VORAUSSETZUNGEN

NW100 oder NW150 mit einer Revision größer/gleich xx.05.

Auf der SPS ist eine Betriebssystem Version größer/gleich 2.00 notwendig. Für die INA2000 Kommunikation muss das Modul "FBPB.BR" gebrannt werden. Das Profibus Konfigurationsmodul ist bereits Bestandteil des Betriebssystems und kann ggf. durch geeignete Tools geändert werden.

Um eine INA2000 Kommunikation über Profibus zu erlauben, muss im SYSCONF Modul die Konfiguration für die INA2000 Profibus Kommunikation erweitert werden.

LS251 KOMMUNIKATION

PC VORAUSSETZUNGEN

Unter Windows wird die LS251 Karte nach dem Einstecken in das System vom Betriebssystem automatisch erkannt. Im Hardware Wizard kann der entsprechende Treiber unter "Pvi\Drivers\Ls251\W2k_xx" ausgewählt werden.

Runtime SPS Voraussetzungen:

Auf der LS251 muss mindestens die SPS Betriebssystemversion V2.01 installiert sein.

AR000 KOMMUNIKATION

Die Kommunikation mit AR000 erfolgt über ein Ethernet UDP Gerät und eine Ethernet UDP Station mit der lokalen IP-Adresse 127.0.0.1.

Anschlussbeschreibung für Gerät: "/IF=TcpIp", Anschlussbeschreibung für Station: "/DAIP=127.0.0.1 /REPO=11160".

PC VORAUSSETZUNGEN

Die AR000 Runtime Emulation muss installiert sein.

Device Parameter - INA2000 Geräteobjekt

Mit dem INA2000 Geräteobjekt wird das verwendete Kommunikationsgerät festgelegt. Die hier beschriebenen Parameter müssen Sie in der Verbindungskonfiguration im Feld **Device Parameter** eingeben.
Folgende Kommunikationsgeräte können definiert werden:

- Serielles Gerät
- CAN Gerät
- Ethernet UDP Gerät
- Modem Gerät
- Profibus FDL Gerät
- LS251 Gerät

Das Kommunikationsgerät wird einschließlich der erforderlichen Geräteparameter in der Anschlussbeschreibung des Geräteobjekts definiert. Innerhalb der Anschlussbeschreibung werden die einzelnen Parameter wieder durch Parameterkennungen unterschieden. Die Parameterkennungen beginnen immer mit dem Zeichen '/'.

SYNTAX DER ANSCHLUSSBESCHREIBUNG

/IF=<Gerätename> [/<Kennung1>=<Parameterwert> [/<Kennung2>=<Parameterwert> ...]]

Der Parameter /IF ist allen Kommunikationsgeräten gemeinsam. Alle anderen Parameter sind abhängig vom verwendeten Kommunikationsgerät. Im Gerätenamen wird zwischen Groß- und Kleinbuchstaben nicht unterschieden. Zwischen den einzelnen Parameterangaben ist mindestens ein Leerzeichen einzufügen.

In der Objektbeschreibung muss die Anschlussbeschreibung immer unter Hochkomma ("...") angegeben werden.

BEISPIEL FÜR ANSCHLUSSBESCHREIBUNG

CD="/IF=com1 /BD=57600 /PA=2"

SERIELLES GERÄT

Nachstehende Tabelle zeigt alle, in der Anschlussbeschreibung des Geräteobjekts definierbaren Parameter für die serielle Kommunikation.

Parameter	Werte	Vorgabe	Beschreibung
/IF	com1 ...comX	keine	Standardmäßig: com1 bis com4. Mit entsprechenden seriellen Schnittstellenkarten oder Adapter ist auch mehr als com4 möglich. Beispiel: "/IF=com1".
/BD	9600, 19200, 38400, 57600, 115200	57600	Baudrate in Bits pro Sekunde.
/RS	-1, 0, 232, 422, 485	232	RS232 oder RS422 Protokoll einstellen. RS485 wird von INA2000 nicht unterstützt. Beispiel: "/RS=422". Mit diesem Parameter wird die Flusskontrolle (Flow Control) der Leitungen CTS (clear-to-send) und RTS (request-to-send) der seriellen Schnittstelle eingestellt. Mögliche Parameterwerte: = -1: PVI verändert nicht die aktuelle Einstellung (siehe Eigenschaften Schnittstellengerät) der CTS/RTS Flusskontrolle. = 0: RTS Leitung immer ausschalten (RTS Flusskontrolle ist deaktiviert). = 232: RTS Handshake verwenden. = 422: RTS Leitung immer einschalten. = 485: RTS Leitung zum Einleiten eines Transfers und während des Transfers einschalten. Die CTS Flusskontrolle ist bei allen Parameterwerten >= 0 deaktiviert. Manche USB/Seriell Adapter (USB auf serielle Schnittstelle) können die CTS/RTS Flusskontrolle nicht korrekt behandeln. Um eine RS232-Verbindung über diese Adapter herzustellen, muss der Parameter "/RS=0" angegeben werden.
/PA	0 - 4	1	Einstellung für Parity. 0=NOPARITY, 1=ODDPARITY, 2=EVENPARITY, 3=MARKPARITY, 4=SPACEPARITY. Standardeinstellung für INA2000 Kommunikation ist "/PA=2".

Beispiel

Beispiel für Anschlussbeschreibung:

CD="/IF=com2 /BD=115200 /PA=2"

CAN GERÄT

Um ein INACAN Gerät ansprechen zu können, muss dieses zuerst mit dem B&R CAN Gerätekonfigurator als CAN Gerät eingerichtet werden. Der Konfigurator wird über die Systemsteuerung (Control Panel) und CAN Geräte (CAN Devices) aufgerufen. Die Geräteliste des Konfigurator zeigt alle bereits eingerichteten CAN Geräte. Die Gerätenummer muss auch im PVI Gerätenamen angegeben werden. Das Gerät CAN1 entspricht dabei dem PVI Gerätenamen INACAN1, CAN2 entspricht INACAN2 usw. Mit dem B&R CAN Gerätekonfigurator werden auch die Geräte-Ressourcen (IRQ, Port Adresse usw.) eingestellt.

Nachstehende Tabelle zeigt alle in der Anschlussbeschreibung des Geräteobjekts definierbaren Parameter für die CAN Kommunikation.

Parameter	Werte	Vorgabe	Beschreibung
/IF	inacan1 ... inacanX	keine	CAN Gerät. Das verwendete Gerät muss in der Geräteliste des B&R CAN Gerätekonfigurators eingetragen sein. Beispiel: "/IF=inacan1".
/CNO	0, 1	0	Nummer (Kanal) des CAN-Controllers. Auf der LS172 Karte sind 2 CAN-Controller vorhanden. Mit dem Parameter /CNO wird der gewünschte Controller ausgewählt. Für den Standard CAN-Controller darf kein anderer Wert als 0 (Null) angegeben werden. Beispiele für LS172 Karte: CAN-Bus 1: "/IF=inacan2 /CNO=0", CAN-Bus 2: "/IF=inacan2 /CNO=1".
/IT	0 - 60000, 0 = aus	0	Intervall-Timeout (ms). Definiert die maximale Zeit, die beim Empfangen und Senden eines INA-Frames zwischen zwei CAN Nachrichten (CAN Messages) vergehen darf. Dieser Parameter wird zusammen mit dem Parameter /RT des INA2000 CPU-Objekts zur Erkennung eines Verbindungsabbruchs verwendet. Der Parameter darf nicht kleiner als die Reaktionszeit der Steuerung (5 - 30 ms) zuzüglich einer Reserve von mindestens 25 ms (bei starker Interrupt-Belastung des PCs entsprechend höher), sollte aber kleiner als /RT sein. Ist keine schnelle Erkennung einer Verbindungsunterbrechung gefordert, so kann die Überwachung des Intervall-Timeouts auch ausgeschaltet werden ("/IT=0"). Beispiel: "/IT=80".
/BI	0 - 2047 / 536870911	1598	Basis CAN-ID der INA2000 Kommunikation. Alle Stationen des INA2000 Netzwerks müssen die gleiche Einstellung haben. Beispiel: "/BI=1598".
/MDA	32 - 255	32	Anzahl der maximal möglichen INA2000 Stationen (= höchste Stationsnummer). Alle Stationen des INA2000 Netzwerks müssen die gleiche Einstellung haben. Beispiel: "/MDA=50".
/SA	1 - /MDA	1	Stationsnummer der Quellstation (eigene Station). Die Stationsnummer muss innerhalb des INA2000 Netzwerks eindeutig sein. Beispiel: "/SA=3".
/BD	10000,	*)	Baudrate (Datenrate) am CAN Bus in Bits pro Sekunde. Die Angabe

	20000, 50000, 100000, 125000, 150000, 250000, 500000, 800000, 1000000		kann auch in kBits pro Sekunde erfolgen. Alle Stationen des INA2000 Netzwerks müssen die gleiche Einstellung haben. Beispiel: "/BD=250000" oder "/BD=250".
/CMODE	11, 29	*)	CAN Kommunikation mit 29 Bit Identifier (Extended Frames) oder mit 11 Bit Identifier (Standard Frames). Werden 29 Bit CAN Identifier (Extended Frames) verwendet, können 11 Bit Identifier nicht empfangen oder gesendet werden. Alle Stationen des INA2000 Netzwerks müssen die gleiche Einstellung haben. Beispiel: "/CMODE=29".
/CT	>= 1, 0 = aus	*)	Zykluszeit (in ms), innerhalb der maximal die angegebene Anzahl von CAN Nachrichten (Parameter /MC) gesendet werden. Werte < 20 ms sind nicht sinnvoll. Beispiel: "/CT=20".
/MC	>= 1	*)	Maximale Anzahl von CAN Nachrichten, die innerhalb der angegebenen Zykluszeit (Parameter /CT) gesendet werden. Beispiel: "/MC=15".

*) Die Parameter /BD, /CMODE, /CT und /MC können auch mit dem B&R CAN Gerätekonfigurator eingestellt werden. Sind diese Parameter in der Anschlussbeschreibung nicht angegeben (= empfohlene Methode), so werden die im Konfigurator eingestellten Werte verwendet. Ist einer der Parameter in der Anschlussbeschreibung angegeben, wird die entsprechende Konfiguratoreinstellung überschrieben.

Die Parameter /CT und /MC dienen zur Nachrichtenbegrenzung. Damit kann die Anzahl der gesendeten CAN Nachrichten pro Zykluszeit beschränkt werden. Die vom PC ausgehende Belastung auf dem CAN Bus wird dadurch vermindert. Die Nachrichtenbegrenzung ist wichtig, wenn andere Bus-Teilnehmer bei eingestellter Baudrate nur eine gewisse Anzahl von empfangenen CAN Nachrichten verarbeiten können oder um allgemein die Interrupt-Belastung anderer Bus-Teilnehmer zu verringern. Der Nachteil der Nachrichtenbegrenzung ist jedoch eine langsamere CAN Kommunikation.

Für die INA2000 Kommunikation benötigt jede Station 3 CAN-IDs. Die CAN-IDs werden aus der Basis CAN-ID (Parameter /BI), der Stationsnummer (Knotennummer) und der maximalen Anzahl der Stationen (Parameter /MDA) gebildet:

- ID1 (Initiate Request) = <Basis CAN-ID> + <Stationsnummer> - 1
- ID2 (Initiate Response) = <Basis CAN-ID> + <Stationsnummer> - 1 + <maximale Anzahl der Stationen> * 2
- ID3 (Data Segment) = <Basis CAN-ID> + <Stationsnummer> - 1 + <maximale Anzahl der Stationen>

> **Beispiel**
>
> Beispiel für Anschlussbeschreibung:
>
> CD="/IF=inacan3 /CNO=1 /SA=3"

ETHERNET UDP GERÄT

Nachstehende Tabelle zeigt alle in der Anschlussbeschreibung des Geräteobjekts definierbaren Parameter für die Ethernet UDP Kommunikation.

Parameter	Werte	Vorgabe	Beschreibung
/IF	tcpip	keine	Ethernet UDP Gerät. Angabe: "/IF=tcpip".
/LOPO	1024 - 32767	11159	Portnummer der Quellstation (eigene Station). Wird der Wert in hexadezimal angegeben, muss ein "0x" vorangestellt werden (z. B. "/LOPO=0x2b97"). Eine andere Portnummer als die Vorgabe muss nur dann eingestellt werden, wenn diese innerhalb des lokalen Rechners nicht eindeutig ist. Beispiel: "/LOPO=11159".
/SA	0 - 99 / 255	1	Stationsnummer der Quellstation (eigene Station). Die Stationsnummer muss innerhalb des INA2000 Netzwerks eindeutig sein. Die höchste Stationsnummer ist 99 bei SG3 und 255 bei SG4. Beispiel: "/SA=3".

Beispiel

Beispiel für Anschlussbeschreibung:

CD="/IF=tcpip /SA=3"

MODEM GERÄT

Nachstehende Tabelle zeigt alle in der Anschlussbeschreibung des Geräteobjekts definierbaren Parameter für die Modem Kommunikation.

Parameter	Werte	Vorgabe	Beschreibung
/IF	modem1 ... modemX	keine	Modem Gerät. Beispiel: "/IF=modem1".
/MO	Modem-Bezeichnung	keine	Bezeichnung des Modems wie im Setup Dialog (Systemsteuerung -> Modem). Der String muss in einfache Hochkommas eingeschlossen werden. Soll innerhalb des Strings ein einfaches Hochkomma verwendet werden, so ist dieses durch zwei einfache Hochkommas darzustellen (z. B. m'56k wird als /MO='m''56k' angegeben). Beispiel: "/MO='MicroLink 56k'".
/TN	Telefon-Nummer	keine	Telefonnummer. Die anzurufende Telefonnummer im Format nach der "ITU-T Recommendation E.123", z. B. +43(7748)6586. Es ist immer die vollständige Telefonnummer anzugeben (mit Landeskennzahl), die Umwandlung in die tatsächliche Telefonnummer geschieht automatisch. Dieser String muss in einfache Hochkommas eingeschlossen werden. Soll auf einen Anruf gewartet werden, ist eine leere Zeichenkette anzugeben (/TN=''). Beispiel: "/TN='+43(7748)6586'".
/MR	0 - INFINITE	INFINITE	Maximale Anzahl erfolgloser Wiederwahlversuche. Gibt an, wie oft versucht werden soll, eine abgebrochene Verbindung wieder aufzubauen. Bei Angabe von INFINITE ist die Anzahl der Wiederwahlversuche unbegrenzt. Ein Wert von 0 bedeutet, dass keine Wiederwahl versucht wird. Der Parameter /MR ist nur von Bedeutung, wenn /TN

			nicht leer ist. Beispiele: "/MR=50", "/MR=INFINITE".
/RI	0 - 3600	60	Zeitintervall zwischen Wiederwahlversuchen in Sekunden. Scheitert ein Wiederwahlversuch, so wird diese Zeit abgewartet, bis mit einem neuen Versuch begonnen wird. Der Parameter /RI ist nur von Bedeutung, wenn /TN nicht leer ist. Beispiel: "/RI=120".
/IT	0 - 60000	40	Intervall-Timeout (ms). Definiert die maximale Zeit, die zwischen dem Empfangen zweier aufeinanderfolgenden Zeichen vergehen darf. Beispiel: "/IT=100".

Neben dem /IF Parameter müssen die Parameter /MO und /TN immer angegeben werden.

Das Lesen von der seriellen Schnittstelle wird von einer Betriebssystemfunktion durchgeführt. Dieser Funktion wird ein Puffer übergeben, in dem die empfangenen Daten abgelegt werden. Das Betriebssystem beendet das Lesen, falls einer der folgenden Situationen eintritt:

1. Der Puffer ist voll.
2. Seit dem Empfang des letzten Zeichens ist ein Zeitspanne größer als das Intervall-Timeout verstrichen.

Die richtige Einstellung des Parameters /IT für das Intervall-Timeout ist daher besonders wichtig. Ist das Intervall-Timeout zu klein, können INA2000 Frames verloren gehen. Dies würde zu Verbindungsabbrüchen führen. Ist das Intervall-Timeout zu groß, leidet der Datendurchsatz, da die Intervall-Timeout Zeit verstreicht, ehe der PC einen empfangenen Frame erkennt. Der Vorgabewert von 40 ms wurde aufgrund von Messungen gewählt, die zeigen, dass ein Modem (durch interne Pufferung) Pausen von ca. 35 ms in den Bytestrom einfügen kann. Im Normalfall sollte es nicht nötig sein, einen anderen Wert zu wählen. Es kann jedoch nicht ausgeschlossen werden, dass bei Verwendung anderer Modemtypen oder schlechter Leitungsqualität (Modems haben eine eigene Übertragungssicherung mit automatischer Sendewiederholung) ein höherer Wert eingestellt werden muss.

Beispiel

Beispiel für Anschlussbeschreibung:

CD="/IF=modem1 /MO='ZyXEL MODEM Omni 288S' /TN='+43(7748)999'"

PROFIBUS FDL GERÄT

Nachstehende Tabelle zeigt alle in der Anschlussbeschreibung des Geräteobjekts definierbaren Parameter für die Profibus FDL Kommunikation.

Parameter	Werte	Vorgabe	Beschreibung
/IF	pbusfdl1 ... pbusfdlX	keine	Profibus Gerät. Beispiel: "/IF=pbusfdl1".
/BA	0x00000 - 0xFFFFF	0xD0000	Basisadresse des DPR der Profibuskarte. Wird der Wert hexadezimal angegeben, muss ein "0x" vorangestellt werden. Beispiel: "/BA=0xD0000".
/FF	Pfadname*)	nw_load.bin	Pfadname der Firmware-Datei für die Profibus-Karte. Beispiel: "/FF=nw.bin".
/FC	Pfadname*)	nw_pb_32.br	Pfadname der Netzwerk Konfigurationsdatei für die Profibus-Karte. Beispiel: "/FC=nw_pb.br".
/CB	1 - 255	2	Anzahl der Kommunikationspuffer auf der Profibus-Karte. Beispiel: "/CB=4".
/SA	0-127	1	Stationsnummer der Quellstation (eigene Station). Die Stationsnummer muss innerhalb des INA2000 Netzwerks eindeutig sein. Beispiel: "/SA=2".

*) Eine Standard Firmware- und Konfigurationsdatei ist Bestandteil des PVI. Wird kein Pfad angegeben, müssen sich die Dateien im Verzeichnis des PVI Managers befinden.

Einstellungen wie Baudrate und Timeout sind nicht über die Geräteparameter zu definieren. Hierfür muss die Konfigurationsdatei mit geeigneten Tools geändert werden.

Beispiel

Beispiel für Anschlussbeschreibung:

CD="/IF=pbusfdl1 /FF=c:\pbconfig\nw_load.bin /FC=c:\pbconfig\nw_pb_32.br /SA=1"

LS251 GERÄT

Nachstehende Tabelle zeigt alle in der Anschlussbeschreibung des Geräteobjekts definierbaren Parameter für die LS251 Kommunikation.

Parameter	Werte	Vorgabe	Beschreibung
/IF	ls251_1 ... ls251_9	keine	LS251 Gerät. Beispiel: "/IF=ls251_1".

Beispiel

Beispiel für Anschlussbeschreibung:

CD="/IF=ls251_1"

6.2.4 Konfigurationsdateien im Redundanzbetrieb

Die PVI Kommunikation benötigt für jede Station (SPS oder Rechner) eine eindeutige Stationsadresse im INA2000 Netzwerk. Das bedeutet im zenon Redundanzbetrieb, dass der Server und der Standby eine unterschiedliche Konfigurationsdatei benötigen. In der Konfigurationsdatei ist für jede Verbindung die eigene Stationsadresse gespeichert.
So erzeugen Sie unterschiedliche Konfigurationsdateien für Server und Standby:

- Richten Sie den Treiber im Editor so ein, dass die Verbindungen korrekt sind.
- Testen Sie die Kommunikation in der Runtime am Server.
- Nehmen Sie den Standby einmalig in Betrieb. Der Standby holt sich dabei das gesamte Projekt inklusive der Konfigurationsdatei vom Server. beenden Sie den Standby wieder
- Öffnen Sie die **Remote-Transport** Konfiguration in den Projekteigenschaften.
- Wählen Sie die Zeile, die für die Treiberdateien zustaändig ist (Zeile 8) aus und setzen Sie diese Zeile auf inaktiv. Diese Einstellung sorgt dafür, dass die Änderungen, die Sie am Standby vornehmen nicht durch die Datei vom Server überschrieben wird. Wenn Sie diese Änderung nicht vornehmen wird der Standby bei jedem Start die aktuelle Konfigurationsdatei vom Server holen und seine eigene, lokale Datei, überschreiben.
- Übertragen Sie diese Änderung zum Server und Laden Sie ihn nach oder starten ihn neu.
- Starten Sie den Windows Explorer am Standby

- Navigieren Sie zum Runtime-Ordner des Projects und von dort in die Verzeichnisstruktur: \RT\FILES\zenon\custom\drivers. Dort finden Sie die Treiberkonfigurationsdatei. Öffnen Sie die Datei mit dem Windows Editor (Notepad).
- Für jede verbindung finden Sie den Eintrag DEVICE_PARAM= in der Datei. In diesem Eintrag ist die Stationsadresse gespeichert (/SA=x). Ändern Sie die Stationsadresse auf eine eindeutige Stationsadresse ab. Speichern Sie die Änderungen und schließen Sie die Datei.
- Erzeugen Sie im Editor eine Variable die die Eigenschaft **Nur von Standby Server anfordern** gesetzt hat. Anhand dieser Variable können Sie in der Runtime kontrollieren, ob der Server und der Standby gleichzeitig mit der Steuerung kommunizieren.

> **Achtung**
>
> *Änderungen in der Treibekonfiguration (z. B. geänderte IP-Adresse) im Editor werden nun nicht mehr über den Remote-Transport zum Server übertragen und auch nicht von dort zum Standby. Sie müssen Sich manuell darum kümmern, dass die Änderungen in die Konfigurationsdateien an Server und Standby einfließen.*

7. Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

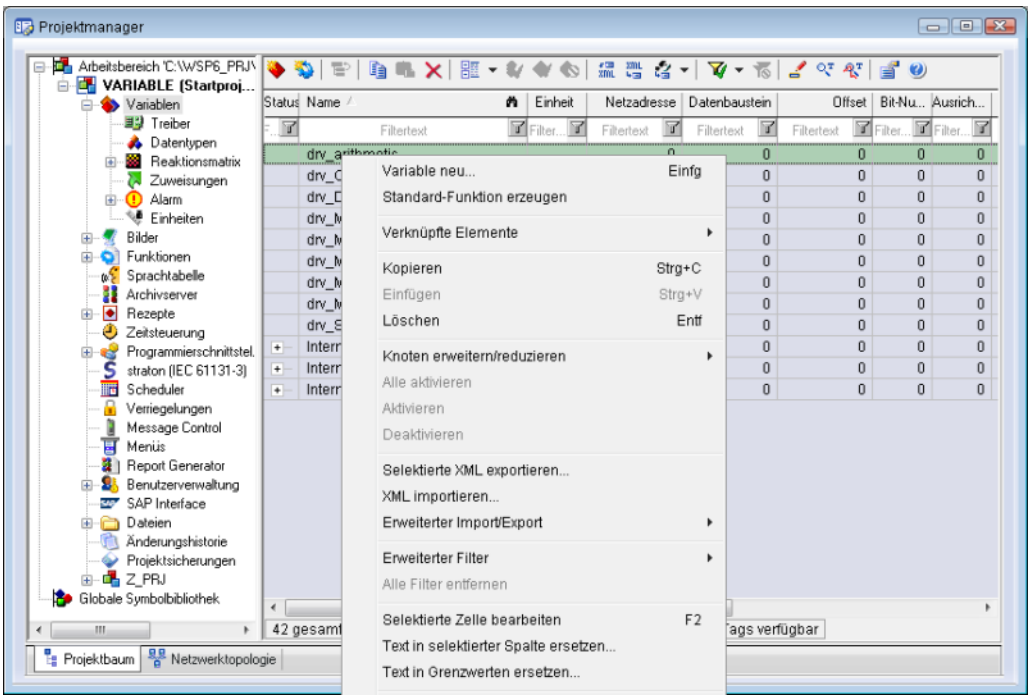
Variablen können angelegt werden:

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

DIALOG VARIABLE

Um eine neue Variable zu erstellen, gleich welchen Typs:

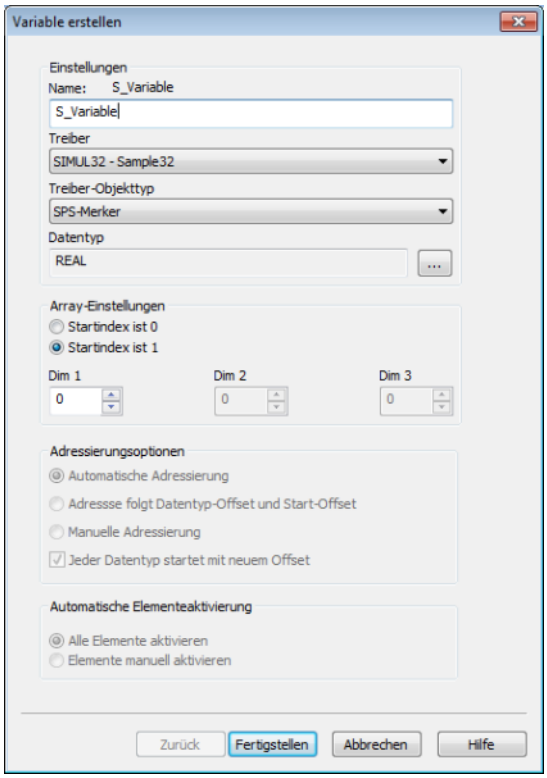
1. Wählen Sie im Knoten **Variablen** im Kontextmenü den Befehl **Variable neu**.



Der Dialog zur Konfiguration der Variable wird geöffnet.

2. Konfigurieren Sie die Variable.

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



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

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

SYMBOLISCHE ADRESSE

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

Maximale Länge: `1024` Zeichen.

ABLEITUNG VOM DATENTYP

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

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

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

7.2 Adressierung

Gruppe/Eigenschaft	Beschreibung
Adressierung	Gruppe mit Eigenschaften für die Adressierung.
Netzadresse	Netzadresse der Variablen. Diese Adresse bezieht sich auf die Netzadresse der Verbindungsprojektierung im Treiber. Damit wird ausgewählt auf welcher Steuerung sich die Variable befindet.
Datenbaustein	Wird für diesen Treiber nicht verwendet.
Offset	Offset der Variablen. Entspricht der Speicheradresse der Variablen in der Steuerung. Einstellbar von 0 bis 4294967295.
Bitnummer	Nummer des Bits innerhalb des eingestellten Offsets. Mögliche Eingabe: 0 bis 65535.
Ausrichtung	Wird für diesen Treiber nicht verwendet.
Stringlänge	Nur verfügbar bei String-Variablen. Maximale Anzahl von Zeichen, die die Variable aufnehmen kann.
PVI Address	Addressinformationen für den B&R PVI Treiber.
PVI Type (VT)	Variablentyp. (Liste der Typen siehe Tabelle: **PVI-Variablentypen**.)
PVI Number of Elements (VN)	Anzahl der Elemente bei Feldvariablen. Vorbelegung: VN=1. Bei mehrdimensionalen Feldvariablen wird die Anzahl der Elemente aller Felddimensionen angegeben (Beispiel: var[10][5] => VN=50).
PVI Length (VL)	Variablenlänge in Bytes. Bei Einzelvariablen entspricht die Variablenlänge der Prozessdatenlänge. Bei Feldvariablen ist die Variablenlänge als Elementlänge definiert.
Treiber Anbindung	*Position der Variable innerhalb des definierten Offsets.*
Treiber	Auswahl des Treibers
Treiberobjekttyp	Objekttyp der Variablen. Wird abhängig vom verwendeten Treiber beim Erstellen der Variablen ausgewählt und kann hier geändert werden.
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.
PVI Name	Wird beim online Import automatisch gesetzt.

> **Achtung**
>
> *Adressierung eindimensionaler Arrays:*
>
> *Werden eindimensionale Arrays mit einem Startindex <> 0 adressiert, muss an den Index ein Beistrich angehängt werden. Details siehe: Dokumentation B&R.*
>
> ***Zum Beispiel:*** `zenon[3,]`

PVI-VARIABLENTYPEN (VT)

Typ	Beschreibung
i8	8 Bit Integer mit Vorzeichen. Variablenlänge: VL=1. Wertebereich: -128 ... 127.
i16	16 Bit Integer mit Vorzeichen. Variablenlänge: VL=2. Wertebereich: -32768 ... 32767.
i32	32 Bit Integer mit Vorzeichen. Variablenlänge: VL=4. Wertebereich: -2147483648 ... 2147483647.
u8	8 Bit Integer ohne Vorzeichen. Variablenlänge: VL=1. Wertebereich: 0 ... 255.
u16	16 Bit Integer ohne Vorzeichen. Variablenlänge: VL=2. Wertebereich: 0 ... 65535.
u32	32 Bit Integer ohne Vorzeichen. Variablenlänge: VL=4. Wertebereich: 0 ... 4294967295.
f32	32 Bit Fließkomma (IEEE Floating). Variablenlänge: VL=4. Wertebereich: -3.402823466e+38 ... -1.175494351e-38 / +1.175494351e-38 ... +3.402823466e+38.
f64	64 Bit Fließkomma (IEEE Floating). Variablenlänge: VL=8. Wertebereich: -1.7976931348623158e+308 ... -2.2250738585072014e-308 / +2.2250738585072014e-308 ... +1.7976931348623158e+308.
boolean	Bit-Variable (Flag) abgebildet auf 1 Byte. Variablenlänge: VL=1. TRUE = Wert ungleich 0, FALSE = Wert gleich 0.
string	String mit 1 Byte Zeichengröße und binär 0 (Null) Abschluss. Die Länge des Stringpuffers ist mit der Variablenlänge (Parameter VL) zu definieren. Die Länge des Stringpuffers ist auch die maximale Stringlänge. Die tatsächliche Stringlänge ist durch das binäre Null-Zeichen bestimmt. Beim Lesen und Schreiben von Prozessdaten mit Variablentyp String ist zu beachten, dass die Daten nur bis einschließlich dem Null-Zeichen übertragen werden. Alle Zeichen nach dem Null-Zeichen sind undefiniert.

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	Beschreibung
In einer Abfrage lesen	36	X	X	`BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING`	Array pollend als ganzes lesen. Details siehe Abschnitt "Geblocktes Lesen von Arrays".
CPU-Status	9	X	--	`BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING`	Status der CPU.
SPS Merker	8	X	X	`UINT,INT`	Datenpunkte pollend/spontan lesen.
Kommunikationsdetails	35	X	X	`BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING`	Variablen für die statische Analyse der Kommunikation; wird zwischen Treiber und Runtime übertragen (nicht zur SPS). **Hinweis**: Die Adressierung und das Verhalten ist bei den meisten zenon Treibern gleich. Weitere Informationen dazu finden Sie im Kapitel Kommunikationsdetails (Treibervariablen) (auf Seite 68).

Legende:

X: wird unterstützt

--: wird nicht unterstützt

GEBLOCKTES LESEN VON ARRAYS

Wenn sich sehr viele Elemente eines Arrays sehr oft ändern, kann das geblockte Lesen von Arrays performanter sein als Kommunikation bei Wertänderung. Dazu können einfache Array-Variablen und Strukturen über den Treiberobjekttyp **In einer Abfrage lesen** geblockt gelesen werden. Ein Block-Array ist

eine große Variable in zenon, die einen kompletten Block von der SPS ausliest. Damit werden viele einzelne Array-Variablen als ein einziger Array gelesen.

Hinweis: Ändern sich im Array immer nur wenige Elemente, ist meistens die Kommunikation im Event-Modus performanter.

Für das geblockte Lesen gilt:

- Es wird immer das komplette Array gepollt. Eine ereignisgesteuerte Kommunikation ist nicht möglich.
- Die Priorität muss für alle Elemente gleich gewählt werden.
- Die PVI-Einstellungen **VN**, **VL** und **VT** werden nur zum Schreiben benötigt und beim Lesen ignoriert. Sie müssen wie bei **SPS Merker** eingestellt sein.
- Einstellungen zum Import werden in den Global Settings (auf Seite 18) der Treiberkonfiguration festgelegt.

> **Achtung**
>
> *Für das geblockte Lesen müssen zenon Datentyp und PVI Typ gleich sein.*
>
> *Also: BOOL -> boolean funktioniert. USINT -> boolean funktioniert nicht.*

EVENT-BETRIEB

Achtung: Die eingestellten Updatezeiten haben auch eine Auswirkung auf die Kommunikation mit der Steuerung im Eventmodus.
Das heißt: Events über Wertänderungen kommen nie öfter als durch die Updatezeit für die Variable festgelegt.

7.3.2 Zuordnung der Datentypen

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

Steuerung	zenon	Datenart
BOOL	BOOL	8
USINT	USINT	9
SINT	SINT	10
UINT	UINT	2
INT	INT	1
UDINT	UDINT	4
DINT	DINT	3
LREAL	LREAL	6
STRING	STRING	12

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

7.4 Variablen anlegen durch Import

Variablen können auch mittels Variablenimport angelegt werden. Für jeden Treiber stehen XML- und DBF-Import zur Verfügung.

> **Info**
>
> *Details zu Import und Export von Variablen finden Sie im Handbuch Import-Export (main.chm::/13028.htm) im Abschnitt Variablen (main.chm::/13045.htm).*

7.4.1 XML Import

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

- `Importieren`:
  Das Element wird neu importiert.
- `Überschreiben`:
  Das Element wird importiert und überschreibt ein bereits vorhandenes Element.

- Nicht importieren:
  Das Element wird nicht importiert.

Hinweis: Beim Import werden die Aktionen und deren Dauer in einem Fortschrittsbalken angezeigt.

VORAUSSETZUNGEN

Beim Import gelten folgende Bedingungen:

- Abwärtskompatibilität

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

- Konsistenz

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

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

- Struktur-Datentypen

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

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

7.4.2 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

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

IMPORT DBF-DATEI

Um den Import zu starten:

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

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

Info

Beachten Sie:

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

EXPORT DBF-DATEI

Um den Export zu starten:

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

Achtung

DBF-Dateien:

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

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

Info

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

DATEIAUFBAU DER DBASE EXPORTDATEI

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

Achtung

dBase unterstützt keine Strukturen oder Arrays (komplexe Variablen).

DBF-Dateien müssen:

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

STRUKTUR

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

LES_SCHR	L	1	Lese-Schreib-Berechtigung 0: Sollwert setzen ist nicht erlaubt 1: Sollwert setzen ist erlaubt
MIT_ZEIT	L	1	Zeitstempelung in zenon (nur wenn vom Treiber unterstützt)
OBJEKT	N	2	Treiberspezifische ID-Nummer des Primitivobjekts setzt sich zusammen aus TREIBER-OBJEKTTYP und DATENTYP
SIGMIN	Float	16	Rohwertsignal minimal (Signalauflösung)
SIGMAX	F	16	Rohwertsignal maximal (Signalauflösung)
ANZMIN	F	16	technischer Wert minimal (Messbereich)
ANZMAX	F	16	technischer Wert maximal (Messbereich)
ANZKOMMA	N	1	Anzahl der Nachkommastellen für die Darstellung der Werte (Messbereich)
UPDATERATE	F	19	Updaterate für Mathematikvariablen (in sec, eine Dezimalstelle möglich) bei allen anderen Variablen nicht verwendet
MEMTIEFE	N	7	Nur aus Kompatibilitätsgründen vorhanden
HDRATE	F	19	HD-Updaterate für hist. Werte (in sec, eine Dezimalstelle möglich)
HDTIEFE	N	7	HD-Eintragtiefe für hist. Werte (Anzahl)
NACHSORT	L	1	HD-Werte als nachsortierte Werte
DRRATE	F	19	Aktualisierung an die Ausgabe (für zenon DDE-Server, in sec, eine Kommastelle möglich)
HYST_PLUS	F	16	Positive Hysterese; ausgehend vom Messbereich
HYST_MINUS	F	16	Negative Hyterese; ausgehend vom Messbereich
PRIOR	N	16	Priorität der Variable
REAMATRIZE	C	32	Name der zugeordnete Reaktionsmatrix
ERSATZWERT	F	16	Ersatzwert; ausgehend vom Messbereich
SOLLMIN	F	16	Sollwertgrenze Minimum; ausgehend vom Messbereich
SOLLMAX	F	16	Sollwertgrenze Maximum; ausgehend vom Messbereich
VOMSTANDBY	L	1	Variable vom Standby Server anfordern; der Wert der Variable wird im redundanten Netzwerkbetrieb nicht vom Server sondern vom Standby Server angefordert
RESOURCE	C	128	Betriebsmittelkennung. Freier String für Export und Anzeige in Listen. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
ADJWVBA	L	1	Nichtlineare Wertanpassung: 0: Nichtlineare Wertanpassung wird verwendet

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

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

GRENZWERTDEFINITION

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

Bezeichnung	Typ	Feldgröße	Bemerkung
AKTIV1	L	1	Grenzwert aktiv (pro Grenzwert vorhanden)
GRENZWERT1	F	20	technischer Wert oder ID-Nummer der verknüpften Variable für einen dynamischen Grenzwert (siehe VARIABLEx) (wenn unter VARIABLEx `1` steht und hier `-1`, wird die bestehende Variablenzuordnung nicht überschrieben)
SCHWWERT1	F	16	Schwellwert für den Grenzwert
HYSTERESE1	F	14	wird nicht verwendet
BLINKEN1	L	1	Blinkattribut setzen
BTB1	L	1	Protokollierung in CEL
ALARM1	L	1	Alarm
DRUCKEN1	L	1	Druckerausgabe (bei CEL oder Alarm)
QUITTIER1	L	1	quittierpflichtig
LOESCHE1	L	1	löschpflichtig
VARIABLE1	L	1	dyn. Grenzwertverknüpfung der Grenzwert wird nicht durch einen absoluten Wert (siehe Feld GRENZWERTx) festgelegt.
FUNC1	L	1	Funktionsverknüpfung
ASK_FUNC1	L	1	Ausführung über die Alarmmeldeliste
FUNC_NR1	N	10	ID-Nummer der verknüpften Funktion (steht hier -1, so wird die bestehende Funktion beim Import nicht überschrieben)
A_GRUPPE1	N	10	Alarm/Ereignis-Gruppe
A_KLASSE1	N	10	Alarm/Ereignis-Klasse
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Farbe als Windowskodierung
GRENZTXT1	C	66	Grenzwerttext
A_DELAY1	N	10	Zeitverzögerung
INVISIBLE1	L	1	Unsichtbar

Bezeichnungen in der Spalte Bemerkung beziehen sich auf die in den Dialogboxen zur Definition von Variablen verwendeten Begriffe. Bei Unklarheiten, siehe Kapitel Variablendefinition.

7.4.3 PVI-Variablen vom Treiber importieren

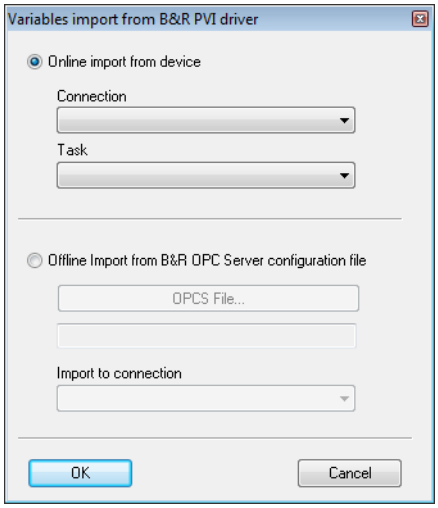
PVI Variablen können offline per OPCS-XML-Datei oder über den Online-Import des Treibers importiert werden.

Beim Import werden die Variablen mit den bestehenden zusammengeführt. Schlüssel dafür ist der Variablenname.

- Variablenbennung
  - Der Variablenname setzt sich zusammen aus: `Verbindungsname.Taskname/Variablenname`
- Adressinformation:
  - Adresse: `Taskname/Variablennname`
  - Datentyp: `u16`
  - Verbindungszuordnung: Netzadresse `0`

Um den Import zu starten:

- wählen Sie im Kontextmenü des Treibers **Variablen vom Treiber importieren...**



Parameter	Beschreibung
Offline Import from B&R OPC Server configuration file	Import über XML-Datei, die der BuR OPC Server generiert.
OPCS File...	Auswahl der OPCS-XML-Datei. Klick auf Schaltfläche öffnet Explorer zur Auswahl der OPCS-Datei.
Import to connection	Auswahl der Verbindung. Klick öffnet Dropdownliste mit allen vorhandenen Verbindungen. Alle ausgewählten XML Variablen werden beim Import über die Netzadresse mit der ausgewählten Verbindung verbunden
Online import from device	Variablen werden Online aus der Steuerung eingelesen.
Connection	Auswahl der Verbindung für den Online Browse.
Task	Auswahl des Tasks, der beim Online Browse ausgelesen wird.

VARIABLEN-ADRESSIERUNG

Die Adressierung erfolgt über die Eigenschaften der angelegten PVI-Variable:

Parameter	Beschreibung
Netzadresse	Bestimmt die Verbindung im Treiber. Referenziert auf die Net Address der Verbindung im Treiberkonfigurationsdialog.
PVI Name	Ist der Name der SPS Variable. Der Name wird aus `Taskname/Variablenname` gebildet.
PVI Type (VT)	Variablentyp. Eine Zusammenstellung aller Variablentypen siehe untenstehende Tabelle.
PVI Number of Elements (VN)	Anzahl der Elemente bei Feldvariablen. Vorbelegung: VN=1. Bei mehrdimensionalen Feldvariablen wird die Anzahl der Elemente aller Felddimensionen angegeben (Beispiel: var[10][5] => VN=50).
PVI Length (VL)	Variablenlänge in Bytes. Bei Einzelvariablen entspricht die Variablenlänge der Prozessdatenlänge. Bei Feldvariablen ist die Variablenlänge als Elementlänge definiert. s

GLOBALE VARIABLEN

Globale Variablen können nur verwendet werden, wenn Sie in einen Task eingebunden sind. Daher werden globale Variablen immer bei den Taskvariablen beim Online Import mit angezeigt und auch als Taskvariablen adressiert: `Taskname/Variablenname`.

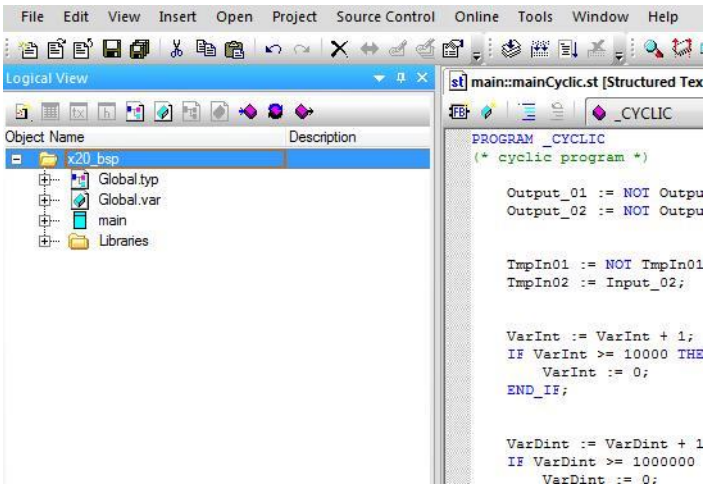
PVI-VARIABLENTYPEN (VT)

Typ	Beschreibung
i8	8 Bit Integer mit Vorzeichen. Variablenlänge: VL=1. Wertebereich: -128 ... 127.
i16	16 Bit Integer mit Vorzeichen. Variablenlänge: VL=2. Wertebereich: -32768 ... 32767.
i32	32 Bit Integer mit Vorzeichen. Variablenlänge: VL=4. Wertebereich: -2147483648 ... 2147483647.
u8	8 Bit Integer ohne Vorzeichen. Variablenlänge: VL=1. Wertebereich: 0 ... 255.
u16	16 Bit Integer ohne Vorzeichen. Variablenlänge: VL=2. Wertebereich: 0 ... 65535.
u32	32 Bit Integer ohne Vorzeichen. Variablenlänge: VL=4. Wertebereich: 0 ... 4294967295.
f32	32 Bit Fließkomma (IEEE Floating). Variablenlänge: VL=4. Wertebereich: -3.402823466e+38 ... -1.175494351e-38 / +1.175494351e-38 ... +3.402823466e+38.
f64	64 Bit Fließkomma (IEEE Floating). Variablenlänge: VL=8. Wertebereich: -1.7976931348623158e+308 ... -2.2250738585072014e-308 / +2.2250738585072014e-308 ... +1.7976931348623158e+308.
boolean	Bit-Variable (Flag) abgebildet auf 1 Byte. Variablenlänge: VL=1. TRUE = Wert ungleich 0, FALSE = Wert gleich 0.
string	String mit 1 Byte Zeichengröße und binär 0 (Null) Abschluss. Die Länge des Stringpuffers ist mit der Variablenlänge (Parameter VL) zu definieren. Die Länge des Stringpuffers ist auch die maximale Stringlänge. Die tatsächliche Stringlänge ist durch das binäre Null-Zeichen bestimmt. Beim Lesen und Schreiben von Prozessdaten mit Variablentyp String ist zu beachten, dass die Daten nur bis einschließlich dem Null-Zeichen übertragen werden. Alle Zeichen nach dem Null-Zeichen sind undefiniert.

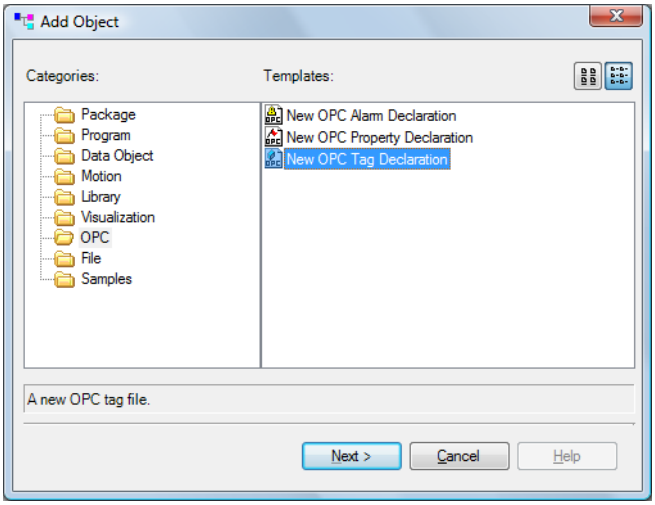
Offline-Import

VORBEREITUNG IM AUTOMATION STUDIO

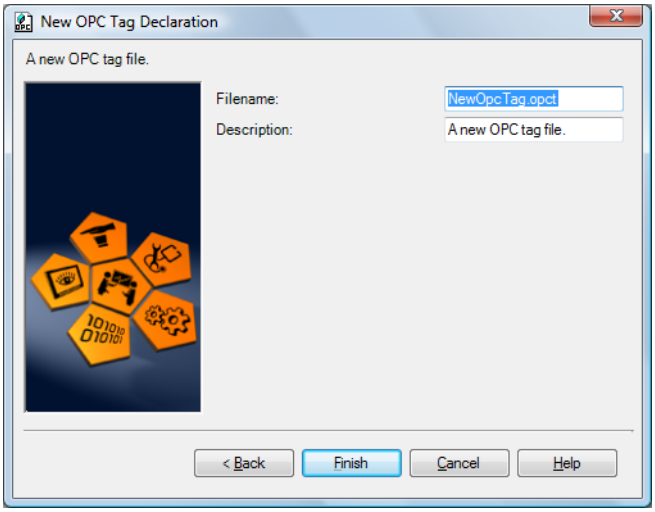
1. navigieren Sie zur gewünschten Datei



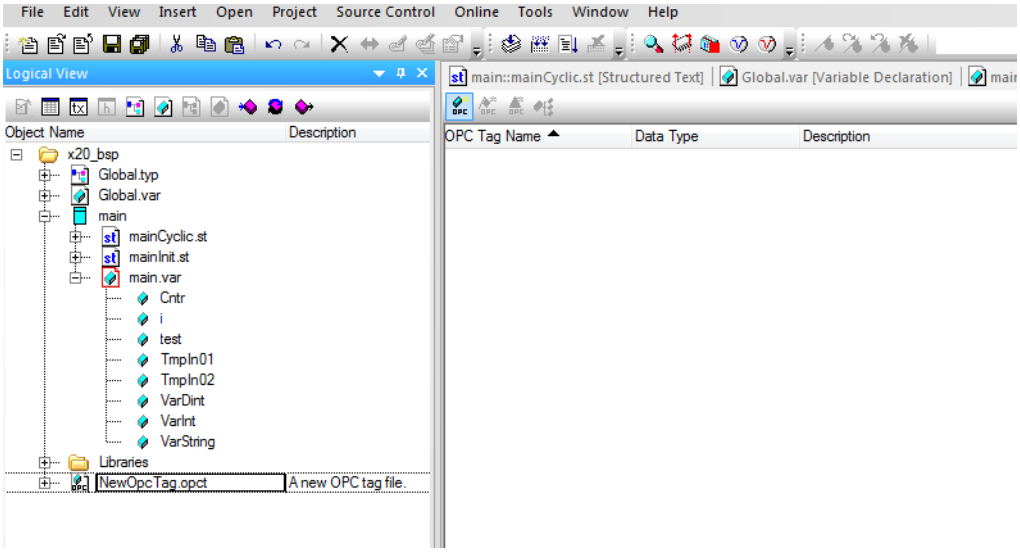
2. wählen Sie im Kontextmenü Add Object...
3. wählen Sie im Ordner **OPC** das Template **New OPC Tag Declaration**



4. vergeben Sie einen Dateinamen mit der Endung .opct

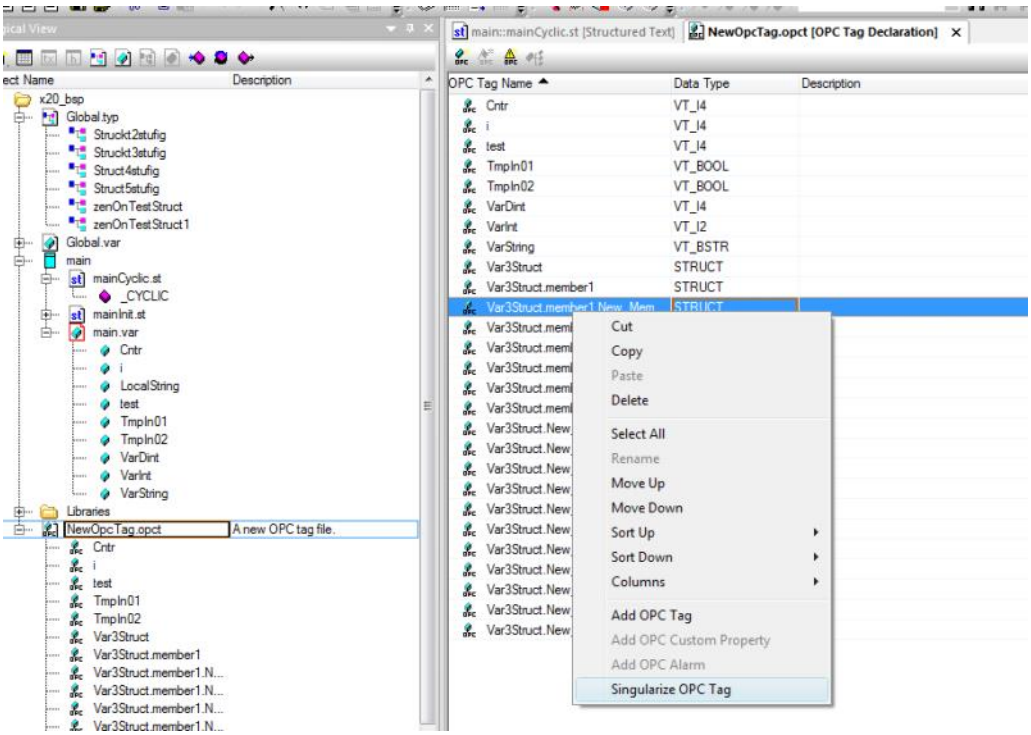


5. öffnen Sie die OPC Taglist im Automation Studio

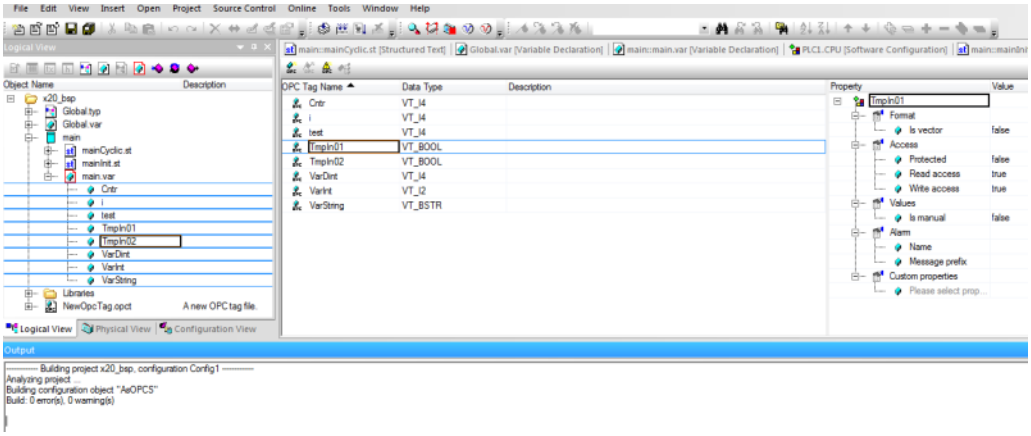


6. fügen Sie per Drag&Drop die Variablen hinzu, die Sie auslesen möchten

7. um Strukturen und Arrays auch offline importieren zu können, müssen die Items der Strukturen aufgelöst werden



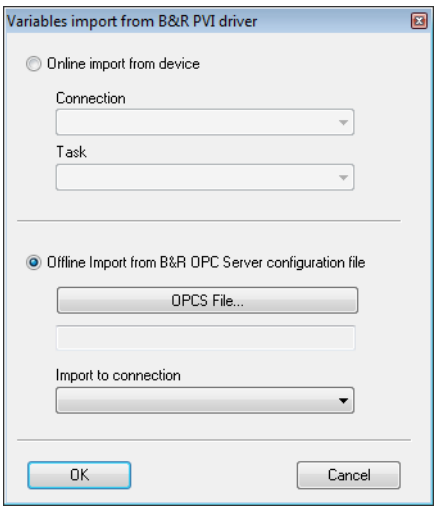
8. beim Kompilieren des Automation Studio Projekts wird die OPCS-Datei erstellt
9. die OPCS-Datei wird im Projektordner von Automation Studio in einem Unterordner abgelegt, zum Beispiel: `\Temp\Objects\Config1\PLC1\AsOPCS.opcs`



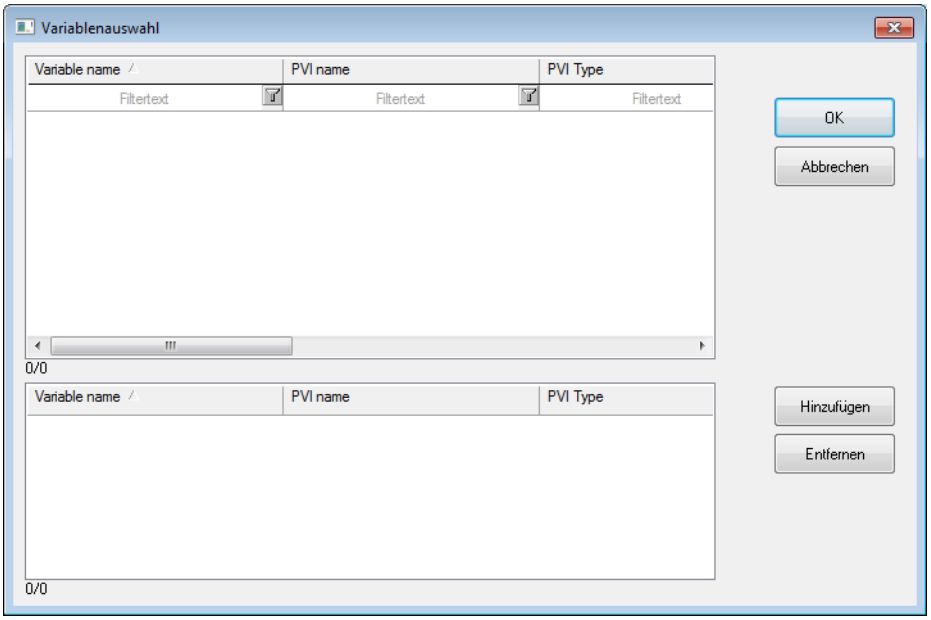
IMPORT IN ZENON

1. wählen Sie im Kontextmenü des Treibers den Befehl **Variablen vom Treiber importieren**
2. wählen Sie im Importdialog **Offline Import from B&R OPC Server configuration file**

3. klicken Sie auf **PPCS File...** und wählen Sie die OPVS-Datei aus

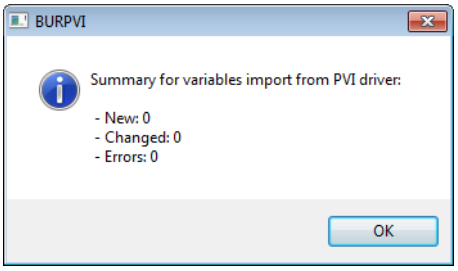


4. wählen Sie die Verbindung, für die importiert werden soll
5. bestätigen Sie den Dialog mit Klick auf OK
6. wählen Sie die gewünschten Variablen mit Doppelklick oder Klick auf die Schaltfläche **Add** aus (Mehrfachauswahl ist möglich)



7. klicken Sie auf die Schaltfläche OK, um die Variablen in zenon zu importieren

8. eine Infobox informiert Sie über importierte Variablen geänderte Variablen und Fehler



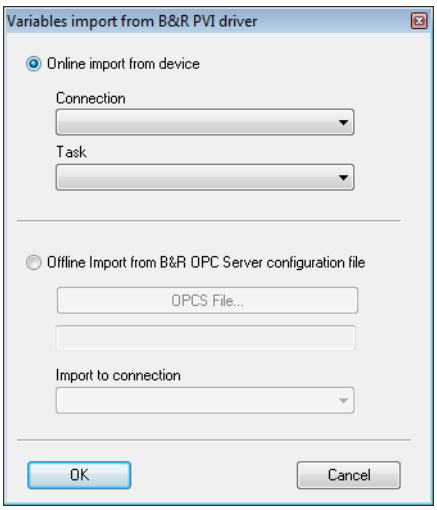
Online-Import

Der Treiber unterstützt im Online Import auch multidimensionale Arrays und Array mit einem Startindex <> 0 und erstellt Variablen, deren Namen dem ursprünglichen Array-Index entsprechen.

> **Achtung**
>
> *Adressierung eindimensionaler Arrays:*
>
> *Werden eindimensionale Arrays mit einem Startindex <> 0 adressiert, muss an den Index ein Beistrich angehängt werden. Details siehe: Dokumentation B&R.*
>
> ***Zum Beispiel:*** `zenon[3,]`

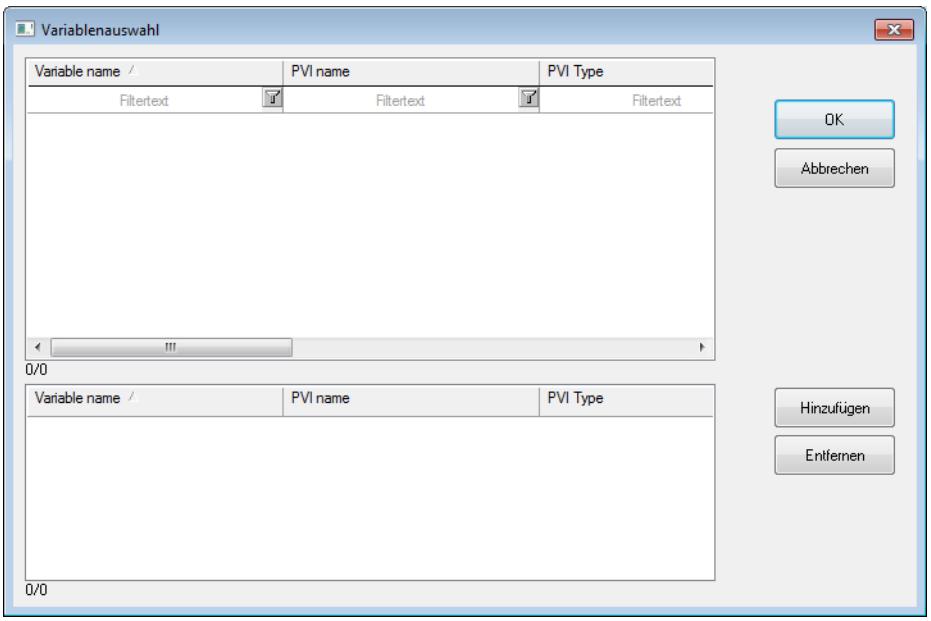
Für den Online-Import:

1. wählen Sie im Kontextmenü des Treibers den Befehl **Variablen vom Treiber importieren**
2. wählen Sie im Importdialog **Online Browse**
3. wählen Sie die Verbindungsart und den gewünschten Task aus

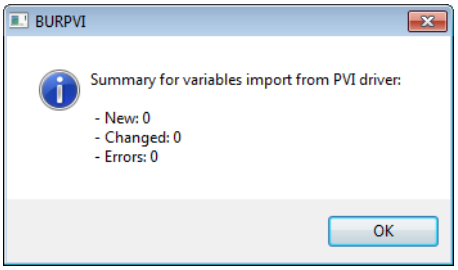


4. bestätigen Sie den Dialog mit Klick auf OK

5. wählen Sie die gewünschten Variablen mit Doppelklick oder Klick auf die Schaltfläche **Add** aus (Mehrfachauswahl ist möglich)



6. klicken Sie auf die Schaltfläche OK, um die Variablen in zenon zu importieren
7. eine Infobox informiert Sie über importierte Variablen, geänderte Variablen und Fehler



7.5 Kommunikationsdetails (Treibervariablen)

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

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

Die Definitionen der im Treiberkit implementierten Variablen sind in der Importdatei **drvvar.dbf** (auf der Installationsmedium im Ordner `\Predefined\Variables`) verfügbar und können von dort importiert werden.

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

Info

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

Zum Beispiel werden:

- Variablen für Modem-Informationen nur von modemfähigen Treibern unterstützt
- 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.

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

KONFIGURATION

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

PhoneNumberSet	STRING	38	Telefonnummer, welche verwendet werden soll.
ModemHwAdrSet	DINT	39	Hardwareadresse, welche zu der Telefonnummer gehört.
GlobalUpdate	UDINT	3	Updatezeit in Millisekunden (ms).
BGlobalUpdaten	BOOL	4	TRUE, wenn die Updatezeit global ist.
TreiberSimul	BOOL	5	TRUE, wenn der Treiber in Simulation ist.
TreiberProzab	BOOL	6	TRUE, wenn das Prozessabbild gehalten werden soll.
ModemActive	BOOL	7	TRUE, wenn das Modem bei diesem Treiber aktiv ist.
Device	STRING	8	Name der seriellen Schnittstelle oder Name des Modem.
ComPort	UINT	9	Nummer der seriellen Schnittstelle.
Baudrate	UDINT	10	Baudrate der seriellen Schnittstelle.
Parity	SINT	11	Parität der seriellen Schnittstelle.
ByteSize	USINT	14	Bitanzahl pro Zeichen der seriellen Schnittstelle. Wert = 0, wenn der Treiber keine serielle Kommunikation herstellen kann.
StopBit	USINT	13	Anzahl der Stoppbits der seriellen Schnittstelle.
Autoconnect	BOOL	16	TRUE, wenn die Modemverbindung automatisch beim Lesen/Schreiben aufgebaut werden soll.
PhoneNumber	STRING	17	Aktuelle Telefonnummer.
ModemHwAdr	DINT	21	Hardwareadresse zur aktuellen Telefonnummer.
RxIdleTime	UINT	18	Wenn länger als diese Zeit in Sekunden (s) erfolgreich kein Datenverkehr stattfindet, wird die Modemverbindung beendet.
WriteTimeout	UDINT	19	Maximale Schreibdauer bei einer Modemverbindung in Millisekunden (ms).

RingCountSet	UDINT	20	So oft läutet ein hereinkommender Anruf, bevor dieser angenommen wird.
ReCallIdleTime	UINT	53	Wartezeit zwischen Anrufen in Sekunden (s).
ConnectTimeout	UINT	54	Zeit in Sekunden (s) für Verbindungsaufbau.

STATISTIK

Name aus Import	Typ	Offset	Erklärung
MaxWriteTime	UDINT	31	Längste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MinWriteTime	UDINT	32	Kürzeste Zeit in Millisekunden (ms), die zum Schreiben benötigt wird.
MaxBlkReadTime	UDINT	40	Längste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
MinBlkReadTime	UDINT	41	Kürzeste Zeit in Millisekunden (ms), die zum Lesen eines Datenblocks benötigt wird.
WriteErrorCount	UDINT	33	Anzahl der Schreibfehler.
ReadSucceedCount	UDINT	35	Anzahl der erfolgreichen Leseversuche.
MaxCycleTime	UDINT	22	Längste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
MinCycleTime	UDINT	23	Kürzeste Zeit in Millisekunden (ms), die zum Lesen aller angeforderten Daten benötigt wurde.
WriteCount	UDINT	26	Anzahl der Schreibversuche.
ReadErrorCount	UDINT	34	Anzahl der fehlerhaften Leseversuche.
MaxUpdateTimeNormal	UDINT	56	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Normal** in Millisekunden (ms).
MaxUpdateTimeHigher	UDINT	57	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höher** in Millisekunden (ms).
MaxUpdateTimeHigh	UDINT	58	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Hoch** in Millisekunden (ms).
MaxUpdateTimeHighest	UDINT	59	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höchste** in Millisekunden (ms).

PokeFinish	BOOL	55	Geht für eine Abfrage auf 1, wenn alle anstehenden Pokes ausgeführt wurden.

FEHLERMELDUNGEN

Name aus Import	Typ	Offset	Erklärung
ErrorTimeDW	UDINT	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler auftrat.
ErrorTimeS	STRING	2	Zeit (in Sekunden seit 1.1.1970), wann der letzte Fehler als String auftrat.
RdErrPrimObj	UDINT	42	Nummer des PrimObjektes, als der letzte Lesefehler verursacht wurde.
RdErrStationsName	STRING	43	Name der Station, als der letzte Lesefehler verursacht wurde.
RdErrBlockCount	UINT	44	Anzahl der zu lesenden Blöcke, als der letzte Lesefehler verursacht wurde.
RdErrHwAdresse	DINT	45	Hardwareadresse, als der letzte Lesefehler verursacht wurde.
RdErrDatablockNo	UDINT	46	Bausteinnummer, als der letzte Lesefehler verursacht wurde.
RdErrMarkerNo	UDINT	47	Merkernummer, als der letzte Lesefehler verursacht wurde.
RdErrSize	UDINT	48	Blockgröße, als der letzte Lesefehler verursacht wurde.
DrvError	USINT	25	Fehlermeldung als Nummer.
DrvErrorMsg	STRING	30	Fehlermeldung als Klartext.
ErrorFile	STRING	15	Name der Fehlerprotokolldatei.

8. Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

- Spar Arrays für Strukturen und Arrays
- im Online Import: multidimensionale Arrays und Array mit einem Startindex <> 0
- Blockwrite

- RDA

RDA

Für die RDA-Archivierung wird ein linearer Speicherbereich in der Steuerung benötigt. Dieser wird über ein Array realisiert. Dieses Array muss den entsprechenden RDA-Header plus Archivdaten enthalten. Es darf nur der Index[0] dieses Arrays aktiviert und als RDA-Variable gekennzeichnet werden. Wird der Array Index[0] auf 1 gesetzt, werden die entsprechenden Werte aus der Steuerung ausgelesen.

Hinweis: Mit "**Nur vom Standby Server anfordern**" konfigurierte Variablen werden in RDA-Archiven nicht unterstützt.

Weiterführende Informationen zur RDA Archivierung finden Sie im zenon Handbuch Archivierung (Archivserver.chm::/28257.htm).

9. Funktion Treiberkommandos

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

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

Achtung: Die zenon Funktion **Treiberkommandos** ist nicht ident mit Treiberkommandos, die bei Energy-Treibern zur Runtime ausgeführt werden können!

Info

Dieses Kapitel beschreibt Standardfunktionalitäten, die für die meisten zenon Treiber gültig sind.
Aber 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.

PROJEKTIERUNG DER FUNKTION

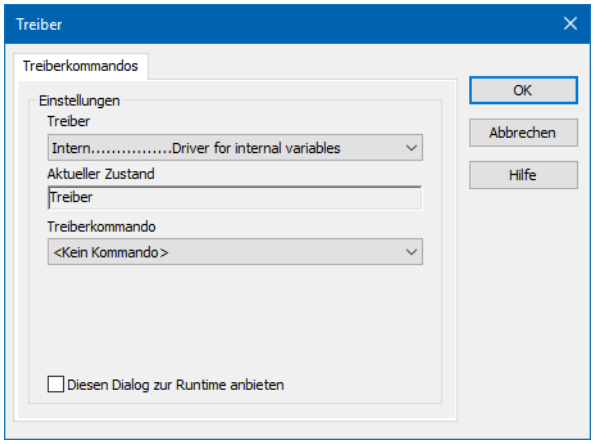
Die Projektierung erfolgt über die Funktion **Treiberkommandos**.
Dazu:

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

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 zur 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, in aktuellen Versionen ohne Funktion.
Treiberkommando	Dropdownliste zur Auswahl des Kommandos:
<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.
`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 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).
`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`	Treiber wird in den zählenden Simulationsmodus gesetzt. Alle Werte werden mit `0` initialisiert.
`Treiber in Simulationsmodus programmiert`	Treiber wird in den zählenden Simulationsmodus gesetzt. 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.
Diesen Dialog zur Runtime anbieten	Dialog wird zur Runtime für Änderungen angeboten.

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, der dann die gewünschte Aktion auf seinem Treiber durchführt. Zusätzlich sendet der Server das gleiche Treiberkommando zum Standby des Projekts. Der Standby führt die Aktion auch auf seinem Treiber aus.

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

10. Fehleranalyse

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

10.1 Analysetool

Alle zenon Module wie z. B. Editor, Runtime, Treiber, usw. schreiben Meldungen in eine gemeinsame LOG-Datei. Um sie korrekt und übersichtlich anzuzeigen, benutzen Sie das Programm Diagnosis Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter *Start/Alle Programme/zenon/Tools 8.00 -> 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

Überprüfen Sie bei Kommunikationsfehler:

- Ist die Steuerung an die Stromversorgung angeschlossen?
- Sind die Teilnehmer im **TCP/IP**-Netz verfügbar?
- Kann die Steuerung über den **Ping** Befehl erreicht werden?
- Kann die Steuerung auf dem entsprechenden Port über **TELNET** erreicht werden?
- Sind Steuerung und PC mit dem passenden Kabel verbunden?
- Wurde der richtige **COM** Port ausgewählt?
- Stimmen die seriellen Kommunikationsparameter (Baudrate, Parität, Start/Stop Bits, ...) überein?
- Wird der **COM** Port durch eine andere Anwendung blockiert?

- Wurde die Netzadresse sowohl im Treiberdialog als auch in den Adresseigenschaften der Variablen korrekt eingestellt.?
- Wird in der Variable der richtige Objekttyp verwendet?
- Stimmt die Offset-Adressierung der Variablen mit der in der Steuerung überein?
- Analyse mit Hilfe des Diagnosis Viewers: Welche Meldungen werden angezeigt?

10.3 Fehlermeldungen

Fehler werden im Ausgabefenster oder in den LOG-Dateien des Dignose Viewers (main.chm::/12464.htm) dokumentiert:

Eintrag	Debug Level	Bedeutung
Block array item XY: Array index [Index] exceeds actual dimenstions [Grenze].	ERROR	Angegebener Index (Adresse) überschreitet die tatsächliche Arraygrenze.
Block array item XY: Incompatible data types. (Data type size [Zahl] differs from actual data type size [Zahl])	ERROR	zenonDatentypen stimmt nicht mit PVI-Typ überein.
Block item XY: Offset '%u' exceeds actual size of the object '%u'.	ERROR	Angegebener Index (Adresse) überschreitet tatsächliche Array-/Strukturgrenzen.
Block item XY: Incompatible data types. (Data type size [Typ] differs from actual data type size [Typ])	ERROR	zenon Datentypen stimmt nicht mit PVI-Typ überein.
Block item XY: Struct element [Index] does not exist.	ERROR	Angegebenes Strukturelement (Adresse) existiert nicht.