zenon Treiber Handbuch

Futurit32

v.7.00

Inhalt

1. Willkommen bei der COPA-DATA Hilfe

ALLGEMEINE HILFE

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

PROJEKTUNTERSTÜTZUNG

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

LIZENZEN UND MODULE

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

2. Futurit32

3. FUTURIT32 - Datenblatt

Allgemein:	
Treiberdateiname	FUTURIT32.exe
Treiberbezeichnung	Swarco Futurit
Steuerungs-Typen	Futurit ACADA 70
Steuerungs-Hersteller	Swarco Futurit;

Treiber unterstützt:	
Protokoll	FuturitCom;
Adressierung: Adress-basiert	x
Adressierung: Namens-basiert	-
Kommunikation spontan	-
Kommunikation pollend	x
Online Browsing	-
Offline Browsing	-
Echtzeitfähig	-
Blockwrite	-
Modemfähig	x
Serielles Logging	-
RDA numerisch	-
RDA String	-

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

Plattformen:	
Betriebssysteme	Windows XP, Vista, 7, Server 2003, Server 2008/R2;
CE Plattformen	-;

4. Treiber-Historie

Datum	Treiberversion	Änderung
07.07.08	100	Treiberdokumentation wurde neu erstellt

5. Voraussetzungen

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

5.1 PC

HARDWARE

Serielle Schnittstelle, Modem

Protokoll: FuturitCom

SOFTWARE

Die Treiber Datei Futurit32.EXE in das aktuelle Installationsverzeichnis kopieren (wenn nicht bereits vorhanden) und ins TREIBER_DE.XML File über das Tool DriverInfo.exe eintragen.

VERBINDUNG

Verbindung von der Seriellen Schnittstelle des PC mit der COM1 Schnittstelle der Futurit ACADA 70. Bei einer Modemanbindung ist am PC ein TAPI-fähiges Modem zu installieren. Installationshinweise hierzu entnehmen Sie bitte der Betriebssystemdokumentation bzw. der Installationsanleitung Ihres Modemherstellers.

5.2 Steuerung

Futurit ACADA 70

Protokoll: FuturitCom

6. Konfiguration

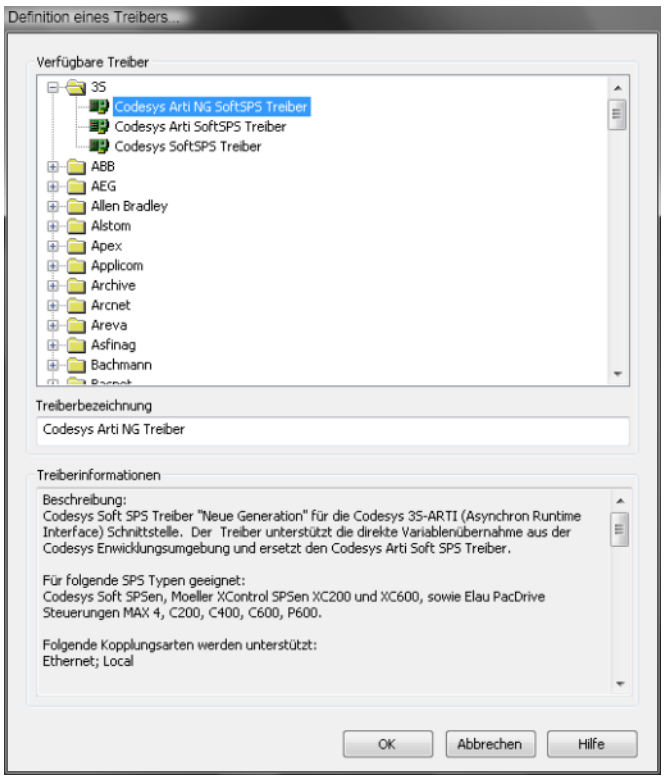
In diesem Kapitel lesen Sie, wie Sie den Treiber im Projekt anlegen und welche Einstellungen beim Treiber möglich sind.

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

6.1 Anlegen eines Treibers

Um einen neuen Treiber anzulegen:

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



- Selektieren Sie den gewünschten Treiber und vergeben Sie eine Bezeichnung für diesen:
  - Die Treiberbezeichnung muss eindeutig sein, d.h. wird ein und derselbe Treiber mehrmals im Projekt verwendet, so muss jeweils eine neue Bezeichnung vergeben werden.
  - Die Treiberbezeichnung ist Bestandteil des Dateinamens. Daher darf Sie nur Zeichen enthalten, die vom Betriebssystem unterstützt werden. Nicht gültige Zeichen werden durch einen Unterstrich (_) ersetzt.

- **Achtung:** Die Bezeichnung kann später nicht mehr geändert werden.

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

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

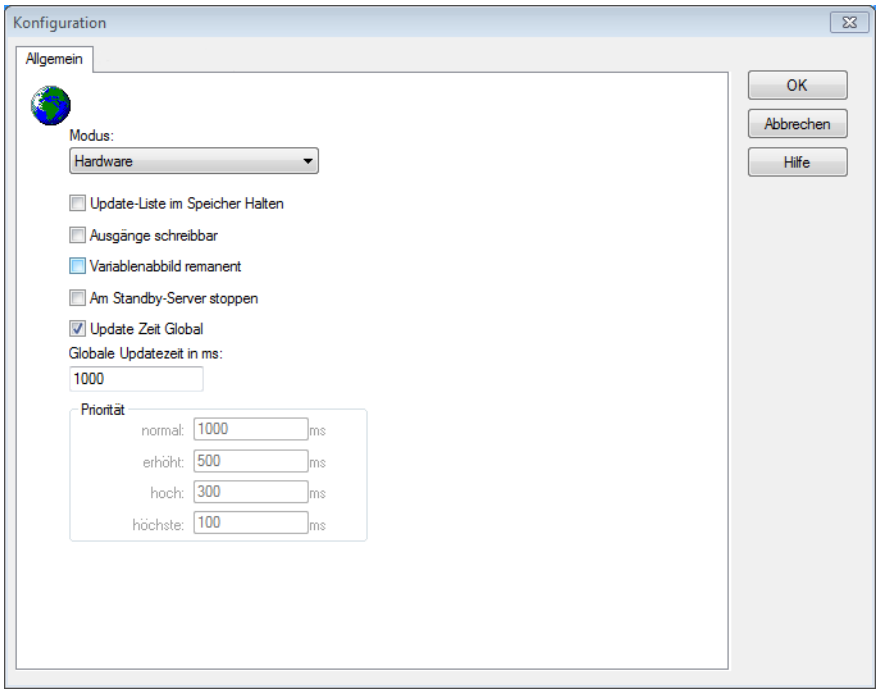
> **Info**
>
> *Bei neu angelegten Projekten oder bei bestehenden Projekten, die in eine Version ab 6.21 konvertiert werden, werden die folgenden Treiber automatisch angelegt:*
>
> ▸ Intern
>
> ▸ MathDr32
>
> ▸ SysDrv.

▸

6.2 Einstellungen im Treiberdialog

Folgende Einstellungen können Sie beim Treiber vornehmen:

6.2.1 Allgemein



Parameter	Beschreibung
Modus	Ermöglicht ein Umschalten zwischen Hardware und Simulationsmodus ▸ Hardware: Die Verbindung zur Steuerung wird hergestellt. ▸ Simulation - statisch Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus bleiben die Werte konstant bzw. die Variablen behalten die über zenon Logic gesetzen Werte. Jede Variable hat seinen eigenen Speicherbereich. z.B. zwei Variablen vom Typ Merker mit Offset 79, können zur Laufzeit unterschiedliche Werte haben und beeinflussen sich gegenseitig nicht. Ausnahme: Der Simulatortreiber. ▸ Simulation - zählend Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden vom Treiber simuliert. In diesem Modus zählt der Treiber die Werte innerhalb ihres Wertebereichs automatisch hoch. ▸ Simulation - programmiert Es wird keine Kommunikation zur Steuerung aufgebaut, die Werte werden von einem frei programmierbaren Simulationsprojekt berechnet. Das Simulationsprojekt wird mit der zenon Logic Workbench erstellt und läuft in einer in den Treiber integrierten zenon Logic Runtime ab. Details siehe Kapitel Treibersimulation. (main.chm::/25206.htm)
Update-Liste im Speicher Halten	Einmal angeforderte Variablen werden weiterhin von der Steuerung angefordert, auch wenn diese aktuell nicht mehr benötigt werden. Dies hat den Vorteil, dass z.B. mehrmalige Bildumschaltungen nach dem erstmaligen Aufschalten beschleunigt werden, da die Variablen nicht neu angefordert werden müssen. Der Nachteil ist eine erhöhte Belastung der Kommunikation zur Steuerung.
Ausgänge schreibbar	Aktiv: Ausgänge können beschrieben werden. Inaktiv: Das Beschreiben der Ausgänge wird unterbunden.

	Hinweis: Steht nicht für jeden Treiber zur Verfügungen.
`Variablenabbild remanent`	Diese Option speichert und restauriert den aktuellen Wert, den Zeitstempel und die Status eines Datenpunkts. Grundvoraussetzung: Die Variable muss einen gültigen Wert und Zeitstempel besitzen. Das Variablenabbild wird im Modus Hardware gespeichert wenn: ▸ einer der Status S_MERKER_1(0) bis S_MERKER8(7), REVISION(9), AUS(20) oder ERSATZWERT(27) aktiv ist Das Variablenabbild wird immer gespeichert wenn: ▸ die Variable vom Objekttyp `Treibervariable` ist ▸ der Treiber im Simulationsmodus läuft. (nicht programmierte Simulation) Folgende Status werden beim Start der Runtime nicht restauriert: ▸ SELECT(8) ▸ WR-ACK(40) ▸ WR-SUC(41) Der Modus **`Simulation - programmiert`** beim Treiberstart ist kein Kriterium, um das remanente Variablenabbild zu restaurieren.
`Am Standby-Server stoppen`	Einstellung für Redundanz bei Treibern, die nur eine Kommunikationsverbindung erlauben. Dazu wird der Treiber am Standby-Server gestoppt und erst beim Hochstufen wieder gestartet. **Achtung:** Ist diese Option aktiv, ist die lückenlose Archivierung nicht mehr gewährleistet. `Aktiv`: Versetzt den Treiber am nicht-prozessführenden Server automatisch in einen Stop-ähnlichen Zustand. Im Unterschied zum Stoppen über Treiberkommando erhält die Variable nicht den Status **`abgeschaltet`** (**`statusverarbeitung.chm::/24150.htm`**), sondern einen leeren Wert. Damit wird verhindert, dass beim Hochstufen zum Server nicht relevante Werte in AML , CEL und Archiv erzeugt werden.
`Update Zeit Global`	`Aktiv`: Die eingestellte `Globale Update Zeit` in `ms` wird für alle Variablen im Projekt verwendet. Die bei den Variablen eingestellte Priorität

	wird nicht verwendet. `Inaktiv`: Die eingestellten Prioritäten werden für die einzelnen Variablen verwendet.
`Priorität`	Hier werden die Pollingzeiten der einzelnen Prioritäten eingestellt. Alle Variablen mit der entsprechenden Priorität werden in der eingestellten Zeit gepollt. Die Zuordnung zu den Variablen erfolgt separat bei jeder Variablen über die Einstellungen in den Variableneigenschaften. Mit den Prioritäten kann die Kommunikation der einzelnen Variablen auf die Wichtigkeit bzw. benötigte Aktualität abgestuft werden. Daraus ergibt sich eine verbesserte Verteilung der Kommunikationslast.

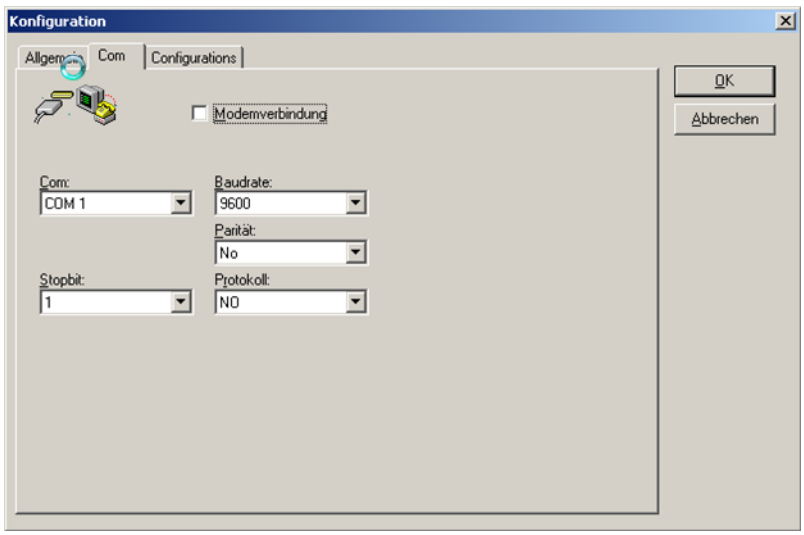
UPDATE ZEIT ZYKLISCHE TREIBER

Für zyklische Treiber gilt:

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

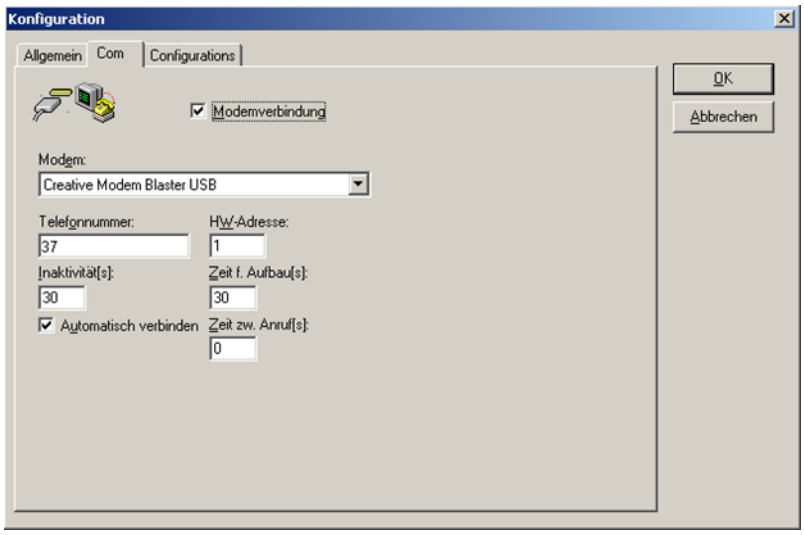
6.2.2 Com

EINSTELLUNGEN



Parameter	Beschreibung
Modemverbindung	Blendet die Felder für die Modemeinstellungen auf (siehe unten).
Com	Auswahl der seriellen Schnittstelle, an der die Steuerung angeschlossen ist
Baudrate	Baudrate der Verbindung (Default: 9600).
Parität	Einstellungen zur Parität der Verbindung (Default: No).
Stopbit	Anzahl der Stopbits der Verbindung (Default: 1).
Protokoll	Protokoll der Verbindung (Default: NO).

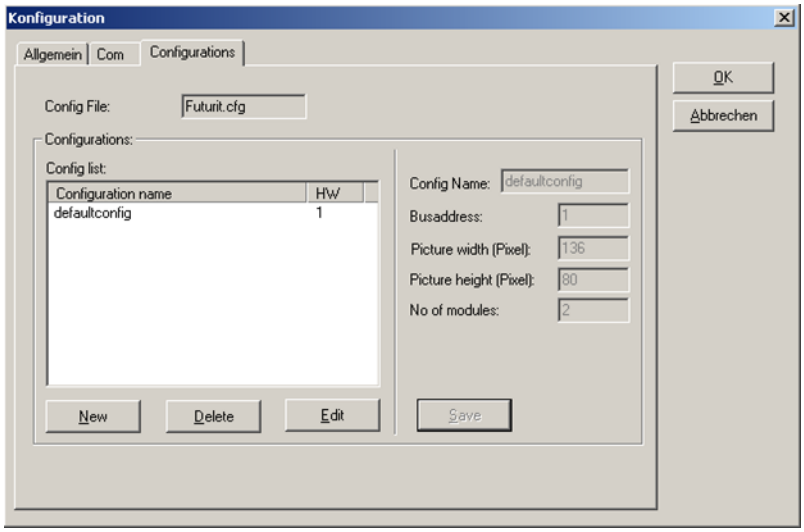
EINSTELLUNGEN – MODEMVERBINDUNG



Parameter	Beschreibung
Modem	Selektieren Sie hier das gewünschte TAPI-Modem.
Telefonnummer	Die Telefonnummer mit welcher eine Verbindung hergestellt werden soll.
HW Adresse	Die Hardware-Adresse für welche die Telefonnummer gilt. Es werden nur Variablen für die definierte Hardwareadresse abgefragt.
Inaktivität	Nach dieser Zeit [s] ohne gültigen Datenaustausch wird die Verbindung beendet.
Zeit für Aufbau	Innerhalb dieser Zeit [s] muss eine Verbindung zustande kommen, ansonsten wird der Verbindungsaufbau mit einem Fehler beendet.
Zeit zwischen Anrufen	Diese Zeit [s] wird zwischen den Anrufen gewartet. Postzugelassene Modem können nicht unmittelbar hintereinander Anrufen.
Automatisch verbinden	Wenn aktiviert dann wird sofort beim Starten der Runtime und vorliegen einer Telefonnummer der Verbindungsaufbau gestartet.

6.2.3 Configurations

Auf dieser Seite können beliebig viele Konfigurationen für Steuerungen auf unterschiedlichen Hardware Adressen gespeichert werden.



Parameter	Beschreibung
Config file	Konfigurationsdatei, in der die Konfiguration gespeichert wird (nur zur Information).
Config List	Zeigt den Konfigurationsnamen mit der dazugehörigen Hardware Adresse an. Durch Anwahl des Verbindungs-namens mit der Maus - Cursor werden die Konfigurationsparameter angezeigt.
Config Name	Frei wählbarer Name der Konfiguration.
Netaddress	Entspricht der Netz Adresse bei der Variablendefinition.
Picture Width (Pixel)	Breite eines Bildes in Pixel.
Picture Height (Pixel)	Höhe eines Bildes in Pixel.
No. of modules	Anzahl der Module.
Speichern	Verbindungseinstellungen speichern.
Neu	Neue Verbindung anlegen. Verbindungsparameter eintragen und mit „Speichern" abschließen.
Löschen	Bestehende Verbindung löschen.
Bearbeiten	Bestehende Verbindung bearbeiten. Verbindungsparameter ändern und mit „Speichern" abschließen.

7. Variablen anlegen

So werden Variablen im zenon Editor angelegt:

7.1 Variablen im Editor anlegen

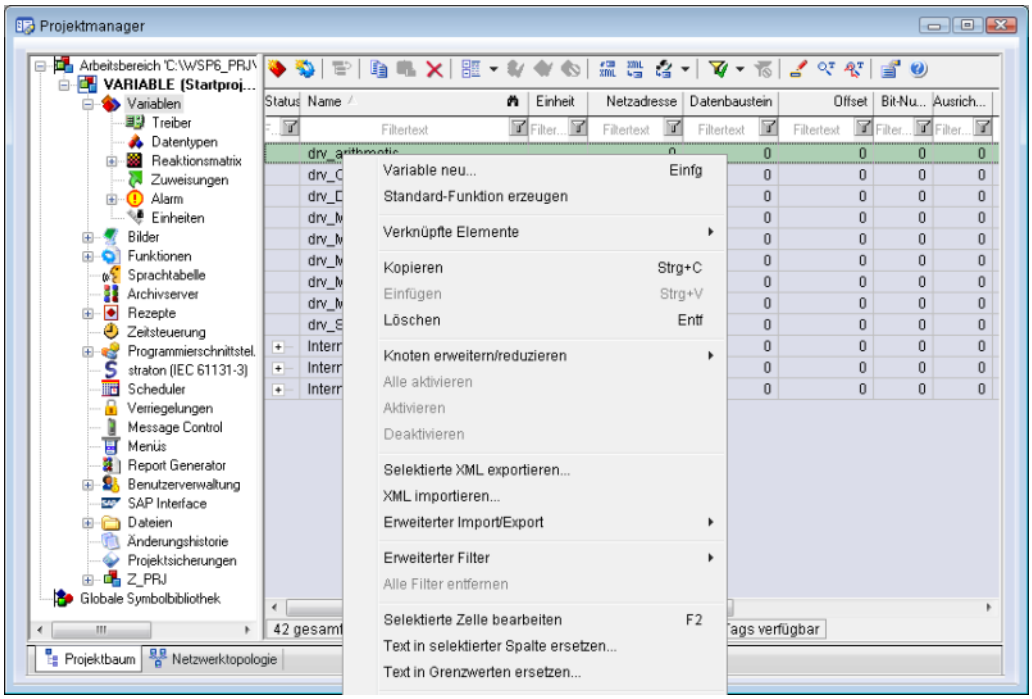
Variablen können angelegt werden:

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

DIALOG VARIABLE

Um eine neue Variable zu erstellen, gleich welchen Typs:

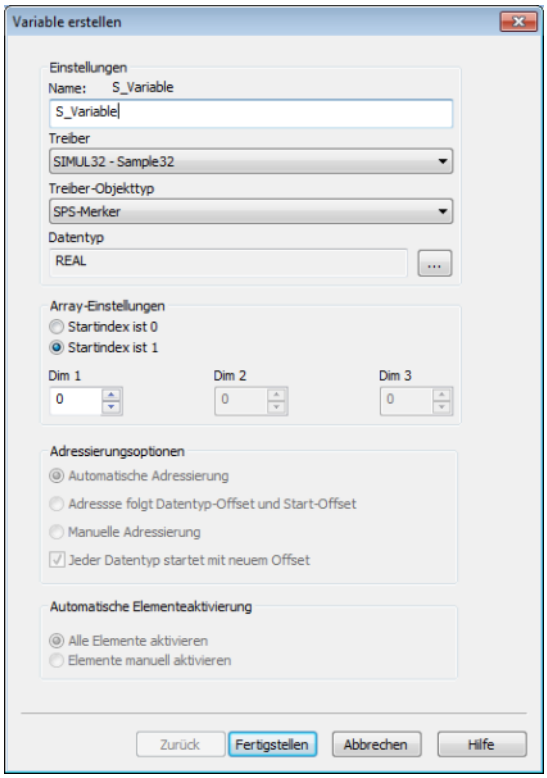
1. wählen Sie im Knoten **`Variablen`** im Kontextmenü den Befehl **`Variable neu`**



2. der Dialog zur Konfiguration der Variable wird geöffnet

3. konfigurieren Sie die Variable

4. welche Einstellungen möglich sind, hängt ab vom Typ der Variablen



Eigenschaft	Beschreibung
Name	Eindeutiger Name der Variablen. Ist eine Variable mit gleichem Namen im Projekt bereits vorhanden, kann keine weitere Variable mit diesem Namen angelegt werden. **Achtung:** Das Zeichen **#** ist im Variablennamen nicht zugelassen. Bei Verwendung nicht zugelassener Zeichen kann die Variablenerstellung nicht abgeschlossen werden, die Schaltfläche **Fertigstellen** bleibt inaktiv.
Treiber	Wählen Sie aus der Dropdownliste den gewünschten Treiber. **Hinweis:** Sollte im Projekt noch kein Treiber angelegt sein, wird automatisch der Treiber für interne Variable (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) geladen.
Treiber-Objekttyp (cti.chm::/28685.htm)	Wählen Sie aus der Dropdownliste den passenden Treiber-Objekttyp aus.

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

ABLEITUNG VOM DATENTYP

Messbereich, Signalbereich und Sollwert Setzen werden immer:

- vom Datentyp abgeleitet
- beim Ändern des Datentyps automatisch angepasst

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

7.2 Adressierung

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

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:

OBJEKTE FÜR PROZESSVARIABLEN IN ZENON

Objekt	Lesen	Schreiben	Kommentar
Konfiguration			Ruft das Treiberkonfigurationsmenü auf
BOOL	J	J	numerische Variable mit 1 Bit
BYTE	J	J	numerische Variable mit 8 Bit
INT	J	J	numerische Variable mit 16 Bit
LINT	J	J	numerische Variable mit 32 Bit
FLOAT	J	J	numerische Variable mit IEEE Format
STRING	J	J	alphanumerische Variable mit bis zu 255 Zeichen

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	8
	USINT	9

	SINT	10
	UINT	2
	INT	1
	UDINT	4
	DINT	3
	ULINT	27
	LINT	26
	REAL	5
	LREAL	6
	STRING	12
	WSTRING	21
	DATE	18
	TIME	17
	DATE_AND_TIME	20
	TOD (Time of Day)	19

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

7.4 Variablen anlegen durch Import

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

7.4.1 XML Import

Für den Import/Export von Variablen gilt:

- Der Import/Export darf nicht aus dem Globalprojekt gestartet werden.

- Der Start erfolgt über:
  - Kontextmenü zu Variablen bzw. Datentyp im Projektbaum
  - oder Kontextmenü einer Variablen bzw. eines Datentyps
  - oder Symbol in der Symbolleiste Variablen

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

7.4.2 DBF Import/Export

Daten können nach dBase exportiert und aus dBase importiert werden.

IMPORT DBF-DATEI

Um den Import zu starten:

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

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

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

EXPORT DBF-DATEI

Um den Export zu starten:

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

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

Das Format der Datei ist im Kapitel Dateiaufbau beschrieben.

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

Dateiaufbau der dBase Exportdatei

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

Achtung

dBase unterstützt keine Strukturen oder Arrays (komplexe Variablen).

DBF-Dateien müssen:

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

STRUKTUR

Bezeichnung	Typ	Feldgröße	Bemerkung
KANALNAME	Char	128	Variablenname. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
KANAL_R	C	128	Ursprünglicher Name einer Variablen, der durch den Eintrag unter KANALNAME ersetzt werden soll (Feld/Spalte muss manuell angelegt werden). Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
KANAL_D	Log	1	Variable wird bei Eintrag 1 gelöscht (Feld/Spalte muss manuell angelegt werden).
TAGNR	C	128	Kennung. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
EINHEIT	C	11	Technische Maßeinheit
DATENART	C	3	Datenart (z.B. Bit, Byte, Wort, ...) entspricht dem Datentyp.
KANALTYP	C	3	Speicherbereich in der SPS (z.B. Merkerbereich, Datenbereich, ...) entspricht Treiber-Objekttyp.
HWKANAL	Num	3	Bus-Adresse
BAUSTEIN	N	3	Datenbaustein-Adresse (nur bei Variablen aus den Datenbereich der SPS)
ADRESSE	N	5	Offset

BITADR	N	2	Für Bit-Variablen: Bitadresse Für Byte-Variablen: 0=niederwertig, 8=höherwertig Für String-Variablen: Stringlänge (max. 63 Zeichen)
ARRAYSIZE	N	16	Anzahl der Variablen im Array für Index-Variablen ACHTUNG: Nur die erste Variable steht voll zur Verfügung. Alle folgenden sind nur über VBA oder den Rezeptgruppen Manager zugänglich
LES_SCHR	L	1	Lese-Schreib-Berechtigung 0: Sollwert setzen ist nicht erlaubt 1: Sollwert setzen ist erlaubt
MIT_ZEIT	L	1	Zeitstempelung in zenon (nur wenn vom Treiber unterstützt)
OBJEKT	N	2	Treiberspezifische ID-Nummer des Primitivobjekts setzt sich zusammen aus KANALTYP und DATENART
SIGMIN	Float	16	Rohwertsignal minimal (Signalauflösung)
SIGMAX	F	16	Rohwertsignal maximal (Signalauflösung)
ANZMIN	F	16	technischer Wert minimal (Messbereich)
ANZMAX	F	16	technischer Wert maximal (Messbereich)
ANZKOMMA	N	1	Anzahl der Nachkommastellen für die Darstellung der Werte (Messbereich)
UPDATERATE	F	19	Updaterate für Mathematikvariablen (in sec, eine Dezimalstelle möglich) bei allen anderen Variablen nicht verwendet
MEMTIEFE	N	7	Nur aus Kompatibilitätsgründen vorhanden
HDRATE	F	19	HD-Updaterate für hist. Werte (in sec, eine Dezimalstelle möglich)
HDTIEFE	N	7	HD-Eintragtiefe für hist. Werte (Anzahl)
NACHSORT	L	1	HD-Werte als nachsortierte Werte
DRRATE	F	19	Aktualisierung an die Ausgabe (für zenon DDE-Server, in sec, eine Kommastelle möglich)
HYST_PLUS	F	16	Positive Hysterese; ausgehend vom Messbereich
HYST_MINUS	F	16	Negative Hyterese; ausgehend vom Messbereich
PRIOR	N	16	Priorität der Variable

REAMATRIZE	C	32	Name der zugeordnete Reaktionsmatrix
ERSATZWERT	F	16	Ersatzwert; ausgehend vom Messbereich
SOLLMIN	F	16	Sollwertgrenze Minimum; ausgehend vom Messbereich
SOLLMAX	F	16	Sollwertgrenze Maximum; ausgehend vom Messbereich
VOMSTANDBY	L	1	Variable vom Standby Server anfordern; der Wert der Variable wird im redundanten Netzwerkbetrieb nicht vom Server sondern vom Standby-Server angefordert
RESOURCE	C	128	Betriebsmittelkennung. Freier String für Export und Anzeige in Listen. Länge kann über den Eintrag MAX_LAENGE in der **project.ini** eingeschränkt werden.
ADJWVBA	L	1	Nichtlineare Wertanpassung: 0: Nichtlineare Wertanpassung wird verwendet 1: Nichtlineare Wertanpassung wird nicht verwendet
ADJZENON	C	128	Verknüpftes VBA-Makro zum Lesen der Variablenwerte für die nichtlineare Wertanpassung.
ADJWVBA	C	128	Verknüpftes VBA-Makro zum Schreiben der Variablenwerte für die nichtlineare Wertanpassung.
ZWREMA	N	16	Verknüpfte Zählwert-Rema.
MAXGRAD	N	16	Maximaler Gradient für die Zählwert-Rema.

Achtung

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

GRENZWERTDEFINITION

Grenzwertdefinition für Grenzwert 1 bis 4, bzw. Zustand 1 bis 4:

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

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

7.5 Treibervariablen

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

- Information
- Konfiguration
- Statistik und
- Fehlermeldungen

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

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

Info

Nicht jeder Treiber unterstützt alle Treibervariablen.

Zum Beispiel werden:

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

INFORMATION

Name aus Import	Typ	Offset	Erklärung
MainVersion	UINT	0	Haupt-Versionsnummer des Treibers.
SubVersion	UINT	1	Sub-Versionsnummer des Treibers.
BuildVersion	UINT	29	Build-Versionsnummer des Treibers.
RTMajor	UINT	49	zenon Hauptversionsnummer
RTMinor	UINT	50	zenon Sub-Versionsnummer
RTSp	UINT	51	zenon Servicepack-Nummer
RTBuild	UINT	52	zenon Buildnummer
LineStateIdle	BOOL	24.0	TRUE, wenn die Modemleitung belegt ist.
LineStateOffering	BOOL	24.1	TRUE, wenn ein Anruf rein kommt.
LineStateAccepted	BOOL	24.2	Der Anruf wird angenommen.
LineStateDialtone	BOOL	24.3	Rufton wurde erkannt.
LineStateDialing	BOOL	24.4	Wahl aktiv.
LineStateRingBack	BOOL	24.5	Während Verbindungsaufbau.
LineStateBusy	BOOL	24.6	Zielstation besetzt.
LineStateSpecialInfo	BOOL	24.7	Spezielle Statusinformation empfangen.
LineStateConnected	BOOL	24.8	Verbindung hergestellt.
LineStateProceeding	BOOL	24.9	Wahl ausgeführt.
LineStateOnHold	BOOL	24.10	Verbindung in Halten.
LineStateConferenced	BOOL	24.11	Verbindung im Konferenzmodus.
LineStateOnHoldPendConf	BOOL	24.12	Verbindung in Halten für Konferenz.
LineStateOnHoldPendTransfer	BOOL	24.13	Verbindung in Halten für Transfer.
LineStateDisconnected	BOOL	24.14	Verbindung beendet.
LineStateUnknow	BOOL	24.15	Verbindungszustand nicht bekannt.
ModemStatus	UDINT	24	Aktueller Modemstatus.
TreiberStop	BOOL	28	Treiber gestoppt

			Bei `Treiberstop`, hat die Variable den Wert `TRUE` und ein **OFF**-Bit. Nach dem Treiberstart, hat die Variable den Wert `FALSE` und kein **OFF**-Bit.
SimulRTState	UDINT	60	Informiert über Status der Runtime bei Treibersimulation.

KONFIGURATION

Name aus Import	Typ	Offset	Erklärung
ReconnectInRead	BOOL	27	Wenn TRUE, dann wird beim Lesen automatisch ein Neuaufbau der Verbindung durchgeführt.
ApplyCom	BOOL	36	Änderungen an den Einstellungen der seriellen Schnittstelle zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyCom zur Folge (aktuell ohne weitere Funktion).
ApplyModem	BOOL	37	Änderungen an den Modemeinstellungen zuweisen. Das Schreiben auf diese Variable hat unmittelbar den Aufruf der Methode SrvDrvVarApplyModem zur Folge. Diese schließt die aktuelle Verbindung und öffnet eine neue entsprechend den Einstellungen **PhoneNumberSet** und **ModemHwAdrSet**.
PhoneNumberSet	STRING	38	Telefonnummer, welche verwendet werden soll.
ModemHwAdrSet	DINT	39	Hardwareadresse, welche zu der Telefonnummer gehört.
GlobalUpdate	UDINT	3	Updatezeit in Millisekunden (ms).
BGlobalUpdaten	BOOL	4	TRUE, wenn die Updatezeit global ist.
TreiberSimul	BOOL	5	TRUE, wenn der Treiber in Simulation ist.

TreiberProzab	BOOL	6	TRUE, wenn das Prozessabbild gehalten werden soll.
ModemActive	BOOL	7	TRUE, wenn das Modem bei diesem Treiber aktiv ist.
Device	STRING	8	Name der seriellen Schnittstelle oder Name des Modem.
ComPort	UINT	9	Nummer der seriellen Schnittstelle.
Baudrate	UDINT	10	Baudrate der seriellen Schnittstelle.
Parity	SINT	11	Parität der seriellen Schnittstelle.
ByteSize	SINT	14	Bitanzahl pro Zeichen der seriellen Schnittstelle. Wert = 0, wenn der Treiber keine serielle Kommunikation herstellen kann.
StopBit	SINT	13	Anzahl der Stoppbits der seriellen Schnittstelle.
Autoconnect	BOOL	16	TRUE, wenn die Modemverbindung automatisch beim Lesen/Schreiben aufgebaut werden soll.
PhoneNumber	STRING	17	Aktuelle Telefonnummer.
ModemHwAdr	DINT	21	Hardwareadresse zur aktuellen Telefonnummer.
RxIdleTime	UINT	18	Wenn länger als diese Zeit in Sekunden (s) erfolgreich kein Datenverkehr stattfindet, wird die Modemverbindung beendet.
WriteTimeout	UDINT	19	Maximale Schreibdauer bei einer Modemverbindung in Millisekunden (ms).
RingCountSet	UDINT	20	So oft läutet ein hereinkommender Anruf, bevor dieser angenommen wird.

ReCallIdleTime	UINT	53	Wartezeit zwischen Anrufen in Sekunden (s).
ConnectTimeout	UDINT	54	Zeit in Sekunden (s) für Verbindungsaufbau.

STATISTIK

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

MaxUpdateTimeHighest	UDINT	59	Zeit seit letzter Aktualisierung der Prioritätsgruppe **Höchste** in Millisekunden (ms).
PokeFinish	BOOL	55	Geht für eine Abfrage auf 1, wenn alle anstehenden Pokes ausgeführt wurden.

FEHLERMELDUNGEN

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

8. Treiberspezifische Funktionen

Dieser Treiber unterstützt folgende Funktionen:

INI EINTRÄGE

ZENON6.INI

keine

PROJECT.INI

keine

ABSETZEN VON KOMMANDOS AN DIE STEUERUNG

SCHRITT 1: DATENBAUSTEIN ANLEGEN

Die Kommandos werden im Leitsystem projektiert, indem für das jeweilige Kommando ein Datenbaustein (Sendebaustein) mit der entsprechenden Nummer angelegt wird.

SCHRITT 2: DATENBAUSTEIN MIT ZU SENDENDEN DATEN FULLEN

Anschließend müssen die Datenpunkte des Datenbausteins mit den Daten für das Kommando gefüllt werden.

SCHRITT 3: KOMMANDO AN DIE STEUERUNG SCHICKEN

Um das Kommando an die Steuerung zu schicken, muss im Byte mit Offset 255 des Datenbausteins die Länge des Kommandos (incl. Kommando-Nummer) abgesetzt werden. Im Offset 255 des zugehörigen Exbausteins (Lesebaustein) wird 0 eingetragen, während das Kommando an die Steuerung geschickt und auf die Antwort gewartet wird.

SCHRITT 4: ERGEBNIS AUSWERTEN

War das Absetzen des Kommandos erfolgreich, wird der Wert des Bytes auf Offset 255 auf 0 zurückgesetzt und im Exbaustein (Lesebaustein) mit der selben Nummer wie das Kommando stehen am Offset 0 Status1 und am Offset 1 Status2. Ab Offset 2 werden eventuell empfangene Daten eingetragen.

Im Offset 255 wird 1 eingetragen, wenn das Kommando erfolgreich ausgeführt worden ist. Im Fehlerfall enthält Offset 255 einen Fehlercode (s.u.).

Ist ein Fehler aufgetreten, wird der Wert des Bytes mit Offset 255 nicht zurückgesetzt. Außerdem wird ein Eintrag in der Fehlerdatei (s.u.) des Treibers gemacht und in die Treibervariable mit Offset 1000 wird der zuletzt aufgetreten Fehler eingetragen.

BEISPIEL

Um zB das Kommando Start mess an die Steuerung zu schicken müssen folgende Schritte durchgeführt werden:

Eine Bytevariable im Datenbaustein (Sendebaustein) $12, also dezimal 18 , Offset 0 anlegen – hier wird die Bildnummer eingetragen

Eine Bytevariable im Datenbaustein (Sendebaustein) $12, also dezimal 18, Offset 255 anlegen – hier wird die Länge des Kommandos eingetragen, um das Kommando abzusetzen

Die Variablen mit Dynelementen verknüpfen, so dass Sollwerte auf die Variablen gesetzt werden können.

In der Runtime zunächst die Variable mit Offset 0 setzen (zB auf 1 für Bild 1)

Anschliessend die Variable mit Offset 255 auf 2 setzen (Das Kommando ist 2 Byte lang).

Hat das funktioniert, wird der Wert der Variable auf Offset 255 auf 0 gesetzt, im Fehlerfall bleibt 2 eingetragen.

Ausserdem kann im Offset 255 des zugehörigen Exbausteins (Lesebaustein) der Fehlercode abgelesen werden, oder 1 bei Erfolg.

SPEZIELLE KOMMANDOS

Die folgenden Kommandos werden speziell behandelt: Send picture, Get picture, Get mess, Led status

SEND PICTURE

Wird Send picture abgesetzt, so wird aus dem Verzeichnis '[Projektverzeichnis]\Bitmaps\' die Datei [Hardware-Adresse]_[Bildnummer].TX zu laden versucht. Die Bildnummer wird dabei aus dem Offset 0 des Datenbausteins (Sendebaustein) $11 (dez. 17) entnommen, der entsprechen projektiert sein muss. Diese wird dann zeilenweise an die Steuerung geschickt. Die Breite und die Höhe des Bildes wird aus der Treiberkonfiguration entnommen. Das Kommando wird ebenfalls erst abgesetzt, wenn im Offset 255 des Datenbausteins 17 die Länge gesetzt wird.

Existiert die Datei nicht, so wird das Kommando normal abgesetzt, dh mit den Informationen aus dem Datenbaustein.

GET PICTURE

Bei Get picture wird ebenfalls die Bildnummer aus Offset 0 von Datenbaustein (Sendebaustein) $1E (dez. 30) entnommen. Das Bild wird dann zeilenweise gelesen, die Anzahl der Zeilen ergibt sich wieder aus der Treiberkonfiguration, und in eine Datei der Form [Hardware-Adresse]_[Bildnummer].TX im Verzeichnis '[Projektverzeichnis]\Bitmaps\Upload' abgelegt.

GET MESS

Bei Get mess wird das Bild zeilenweise gelesen, die Anzahl der Zeilen ergibt sich wieder aus der Treiberkonfiguration. Es wird dann in eine Datei der Form [Hardware-Adresse]_ACT.TX im Verzeichnis '[Projektverzeichnis]\Bitmaps\Upload' abgelegt.

LED STATUS

Bei Led status wird das Bild ebenfalls zeilenweise gelesen, die Anzahl der Zeilen ergibt sich wieder aus der Treiberkonfiguration. Es wird dann in eine Datei der Form [Hardware-Adresse]_ERR.TX im Verzeichnis '[Projektverzeichnis]\Bitmaps\Upload' abgelegt.

ZYKLISCHE AUSFÜHRUNG VON KOMMANDOS

Um Kommandos zyklisch auszuführen muss im Datenbaustein (Sendebaustein) des jeweiligen Kommandos auf Offset 254 die Update-Zeit in Sekunden eingegeben werden. Bei 0 wird normal gelesen (s.o.). Bei einer Zeit ungleich 0 muss anschliessend die Länge des Kommandos richtig gesetzt werden. Ab diesem Zeitpunkt wird das Kommando immer wieder im Abstand von der in Offset 254 übergebenen Sekundenintervallen ausgeführt.

BEISPIEL

Um zB das Kommando Start mess zyklisch auszuführen müssen folgende Schritte durchgeführt werden:

Eine Bytevariable im Datenbaustein (Sendebaustein) $12, also dezimal 18 , Offset 0 anlegen – hier wird die Bildnummer eingetragen

Eine Bytevariable im Datenbaustein (Sendebaustein) $12, also dezimal 18, Offset 254 anlegen – hier wird das Updateintervall in Sekunden eingetragen

Eine Bytevariable im Datenbaustein (Sendebaustein) $12, also dezimal 18, Offset 255 anlegen – hier wird die Länge des Kommandos eingetragen, um das Kommando abzusetzen

Die Variablen mit Dynelementen verknüpfen, so dass Sollwerte auf die Variablen gesetzt werden können.

In der Runtime zunächst die Variable mit Offset 0 setzen (zB auf 1 für Bild 1)

Anschliessend die Variable mit Offset 254 auf zB 5 setzen (für 5 Sekunden Abfrageintervall).

Schliesslich die Variable mit Offset 255 auf 2 setzen (Das Kommando ist 2 Byte lang).

Die Länge des Kommandos auf Offset 255 wird in keinem Fall zurückgesetzt.

Im Offset 255 des zugehörigen Exbausteins (Lesebaustein) kann der Fehlercode abgelesen werden, oder 1 bei Erfolg.

Die speziellen Kommandos funktionieren ebenso wie im normalen Modus auch beim zyklischen Abfragen.

DATENBAUSTEIN $10 (DEZIMAL 16) – DAS KOMMANDO POLL

Wird ein Kommando an die Steuerung geschickt, so enthält die Antwort zwei Statusbytes. Diese werden immer auf Offset 0 und 1 des Exbausteins (Lesebausteins) $10 kopiert.

GESAMTSTATUS

Der Gesamtstatus der Kommandos wird in Exbaustein (Lesebaustein) 0 auf Offset 255 gesetzt.

Der Status ist 0, wenn mindestens ein Kommando noch nicht erfolgreich ausgeführt worden ist.

Erst wenn alle Kommandos erfolgreich waren, wird der Gesamtstatus auf 1 gesetzt.

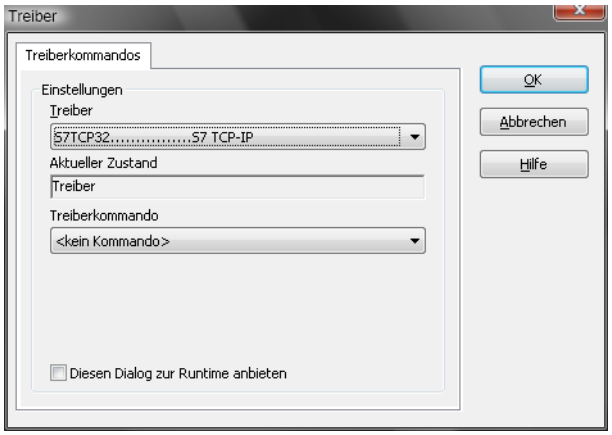
Berücksichtigt werden dabei alle Kommandos, die zu einer Hardware-Adresse projektiert worden sind, dh jedes Kommando, zu dem ein Datenbaustein angelegt wurde, also eine Variable in dem Datenbaustein projektiert und diese in der Runtime abgefragt wurde. In der Treibervariable 1001 wird dieser Status für die aktuelle Modem Hardware-Adresse ebenfalls abgelegt

9. Treiberkommandos

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

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

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



Parameter	Beschreibung
`Treiber`	Dropdownliste mit allen im Projekt geladenen Treibern.
`Aktueller Zustand`	Fixer Eintrag, in aktuellen Versionen ohne Funktion.
`Treiberkommando`	Dropdownliste zur Auswahl des Kommandos.
▸ `Treiber starten (Online-Modus)`	Treiber wird neu initialisiert und gestartet.
▸ `Treiber stoppen (Offline-Modus)`	Treiber wird angehalten, es werden keine neuen Daten angenommen. **Hinweis:** Ist der Treiber im Offline-Modus, erhalten alle Variablen, die für diesem Treiber angelegt wurden, den Status `Abgeschaltet` (`OFF`; Bit 20).

Treiber in Simulationsmodus	Treiber wird in den Simulationsmodus gesetzt. Die Werte aller Variablen des Treibers werden vom Treiber simuliert. Es werden keine Werte von der angeschlossenen Hardware (z.B. SPS, Bussystem, ...) angezeigt.
Treiber in Hardwaremodus	Treiber wird in den Hardwaremodus gesetzt. Für die Variablen des Treibers werden die Werte von der angeschlossenen Hardware (z.B. SPS, Bussystem, ...) angezeigt.
Treiberspezifisches Kommando	Eingabe treiberspezifischer Kommandos. Öffnet Eingabefeld für die Eingabe eines Kommandos.
Treiber Sollwertsetzen aktivieren	Sollwert setzen auf Treiber ist erlaubt.
Treiber Sollwertsetzen deaktivieren	Sollwert setzen auf Treiber wird verhindert.
Verbindung mit Modem aufbauen	Verbindung aufbauen (für Modem-Treiber). Öffnet Eingabefelder für Hardware-Adresse und Eingabe der zu wählenden Nummer.
Verbindung mit Modem trennen	Verbindung beenden (für Modem-Treiber).
Diesen Dialog zur Runtime anbieten	Dialog wird zur Runtime für Änderungen angeboten.

TREIBERKOMMANDOS IM NETZWERK

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

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

10. Fehleranalyse

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

10.1 Analysetool

Alle zenon Module wie z.B. Editor, Runtime, Treiber, usw. schreiben Meldungen in eine gemeinsame Log-Datei. Um sie korrekt und übersichtlich anzuzeigen, benutzen Sie das Programm Diagnose Viewer (main.chm::/12464.htm), das mit zenon mitinstalliert wird. Sie finden es unter *Start/Alle Programme/zenon/Tools 7.00 -> Diagviewer.*

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

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

Im Diagnose Viewer kann man auch:

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

Hinweise:

1. Unter Windows CE werden aus Ressourcegründen auch Fehler standardmäßig nicht protokolliert.
2. Der Diagnose Viewer zeigt alle Einträge in UTC (Koordinierter Weltzeit) an und nicht in der lokalen Zeit.

3. Der Diagnose Viewer zeigt in seiner Standardeinstellung nicht alle Spalten einer Log-Datei an. Um mehr Spalten anzuzeigen, aktivieren Sie die Eigenschaft **`Add all columns with entry`** im Kontextmenü der Spaltentitel.

4. Bei Verwendung von reinem **`Error-Logging`** befindet sich eine Problembeschreibung in der Spalte **`Error text`**. In anderen Diagnose-Ebenen befindet sich diese Beschreibung in der Spalte **`General text`**.

5. Viele Treiber zeichnen bei Kommunikationsprobleme auch Fehlernummern auf, die die SPS ihnen zuweist. Diese werden in **`Error text`** und/oder **`Error code`** und/oder **`Driver error parameter(1`** und **`2)`** angezeigt. Hinweise zur Bedeutung der Fehlercodes erhalten Sie in der Treiberdokumentation und der Protokoll/SPS-Beschreibung.

6. Stellen Sie am Ende Ihrer Tests den Diagnose-Level von **`Debug`** oder **`Deep Debug`** wieder zurück. Bei **`Debug`** und **`Deep Debug`** fallen beim Protokollieren sehr viele Daten an, die auf der Festplatte gespeichert werden und die Leistung Ihres Systems beeinflussen können. Diese werden auch nach dem Schließen des **`Diagnose Viewers`** weiter aufgezeichnet.

Info

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

10.2 Fehlernummern

FEHLERCODES

Fehlercode	Beschreibung
-1	Allgemeiner Fehler
-10	Empfangsdaten: falsche Sync-Bytes
-11	Empfangsdaten: falsche Zieladresse
-12	Empfangsdaten: falsches Tag (nicht zum gesendeten Kommando passend)
-13	Empfangsdaten: Es wurde NACK zurückgegeben
-14	Empfangsdaten: Falsche Cheksumme im Header
-15	Empfangsdaten: Falsche Cheksumme im den Daten
-40	Treiber wurde beendet
-41	Keine Antwort

10.3 Checkliste

Ist das Gerät (die SPS), mit dem versucht wird eine Kommunikation herzustellen, an das Stromversorgungsnetz angeschlossen ?

Sind der PC bzw. die SPS mit einem Nullmodemkabel verbunden ?

Sind die verwendeten Bausteine in der SPS korrekt angelegt ?

Ist die [Treiberbezeichnung].cfg-Datei am Zielrechner?

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

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

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