zenon

by COPA-DATA

zenon driver manual
BeckhNG

v.8.20

Contents

1 Welcome to COPA-DATA help

ZENON VIDEO TUTORIALS

You can find practical examples for project configuration with zenon in our YouTube channel (https://www.copadata.com/tutorial_menu). The tutorials are grouped according to topics and give an initial insight into working with different zenon modules. All tutorials are available in English.

GENERAL HELP

If you cannot find any information you require in this help chapter or can think of anything that you would like added, please send an email to documentation@copadata.com.

PROJECT SUPPORT

You can receive support for any real project you may have from our customer service team, which you can contact via email at support@copadata.com.

LICENSES AND MODULES

If you find that you need other modules or licenses, our staff will be happy to help you. Email sales@copadata.com.

2 BeckhNG

TESTED WITH THE FOLLOWING HARDWARE AND SOFTWARE

TwinCAT SoftSPS 2.9 Build 1031

TESTING ENVIRONMENT

zenon project with one driver and communication to four TwinCAT soft PLCs and a Beckhoff BC bus controller. One TwinCAT soft PLC is running locally on the PC, where the zenon project is running. A second one is running on another PC. The two other TwinCAT soft PLCs are running on two Beckhoff CX1000 CE terminals.

3 BeckhNG - data sheet

General:	
Driver file name	BeckhNG.exe
Driver name	Beckhoff TwinCat NG Driver
PLC types	TwinCAT Soft PLC 2.6 or higher - for Twincat v3 too
PLC manufacturer	Beckhoff

Driver supports:	
Protocol	TC-ADS
Addressing: Address-based	Address based
Addressing: Name-based	--
Spontaneous communication	--
Polling communication	X
Online browsing	X
Offline browsing	X
Real-time capable	--
Blockwrite	--
Modem capable	--
RDA numerical	X
RDA String	--
Hysteresis	--
extended API	X
Supports status bit	X

Driver supports:	
WR-SUC	
alternative IP address	--

Requirements:	
Hardware PC	--
Software PC	The TC-ADS software must be installed. (TC-ADS is on the installation CD). TwinCAT does not need to be installed on the same computer. The use of TwinCAT CP is recommended.
Hardware PLC	TwinCAT Soft PLC on PC and CE
Software PLC	--
Requires v-dll	X

Platforms:	
Operating systems	Windows 10; Windows 7; Windows 8; Windows 8.1; Windows Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

4 Driver history

Date	Driver version	Change
07.07.08	2100	Created driver documentation
11/28/2013	7.11.0.9047	Block arrays (on page 48) are read.

DRIVER VERSIONING

The versioning of the drivers was changed with zenon 7.10. There is a cross-version build number as of this version. This is the number in the 4th position of the file version,
For example: **7.10.0.4228** means: The driver is for version **7.10** service pack **0**, and has the build number **4228**.

Expansions or error rectifications will be incorporated into a build in the future and are then available from the next consecutive build number.

Example

A driver extension was implemented in build **4228**. The driver that you are using is build number **8322**. Because the build number of your driver is higher than the build number of the extension, the extension is included. The version number of the driver (the first three digits of the file version) do not have any significance in relation to this. The drivers are version-agnostic

5 Requirements

This chapter contains information on the requirements that are necessary for use of this driver.

5.1 PC

In order to be able to use the BeckhNG driver with zenon, the following requirements must be met:

- The Beckhoff TwinCat ADS Communication Library must be installed on the computer. To do this, place the **TcAdsDll.dll** file in the zenon program folder. You can find this file on the zenon installation medium.
- The **BeckhNG.EXE** file and the **BeckhNGV.dll-** file must also be present in the zenon program folder.

Ensure that the required files are stored in the corresponding program folder:

- 32-bit operating systems:
  ...\Program Files (x86)\COPA-DATA\zenon 820 SP0
- 64-bit operating systems:
  ...\Programs\COPA-DATA\zenon 820 SP0

The individual PLC stations must be notified of the operating system.

- To do this, define the stations with the **TwinCAT AMS Router**.
- Necessary entries per station:
  - I-address
  - AS NET-ID
  - Port number

- Communication channel

If the **TwinCAT AMS Router** is not used on the computer, a minimal installation of **TWIN CAT (TwinCAT CP)** can be used for the necessary configuration.

Alternatively, the stations can also be notified to the operating system with the **TcAmsRemoteMgr.exe** tool. However, in doing so, it should be noted that it is not the AMS timeout that can be configured in the zenon driver dialog that is used for communication, but the NET-timeout valid in the network. This can result in long error timeouts, if there are connection problems with several stations.

Information

The driver is available for WinCE.

The driver can only be used once for each project in WinCE.

5.2 Control

In the project settings of the TwinCAT PLC Control under "Symbol configuration" both options for creating symbols have to be activated. With this entry, the symbol file (*.tpy) is generated, which is used for the import of the variables in zenon.

6 Configuration

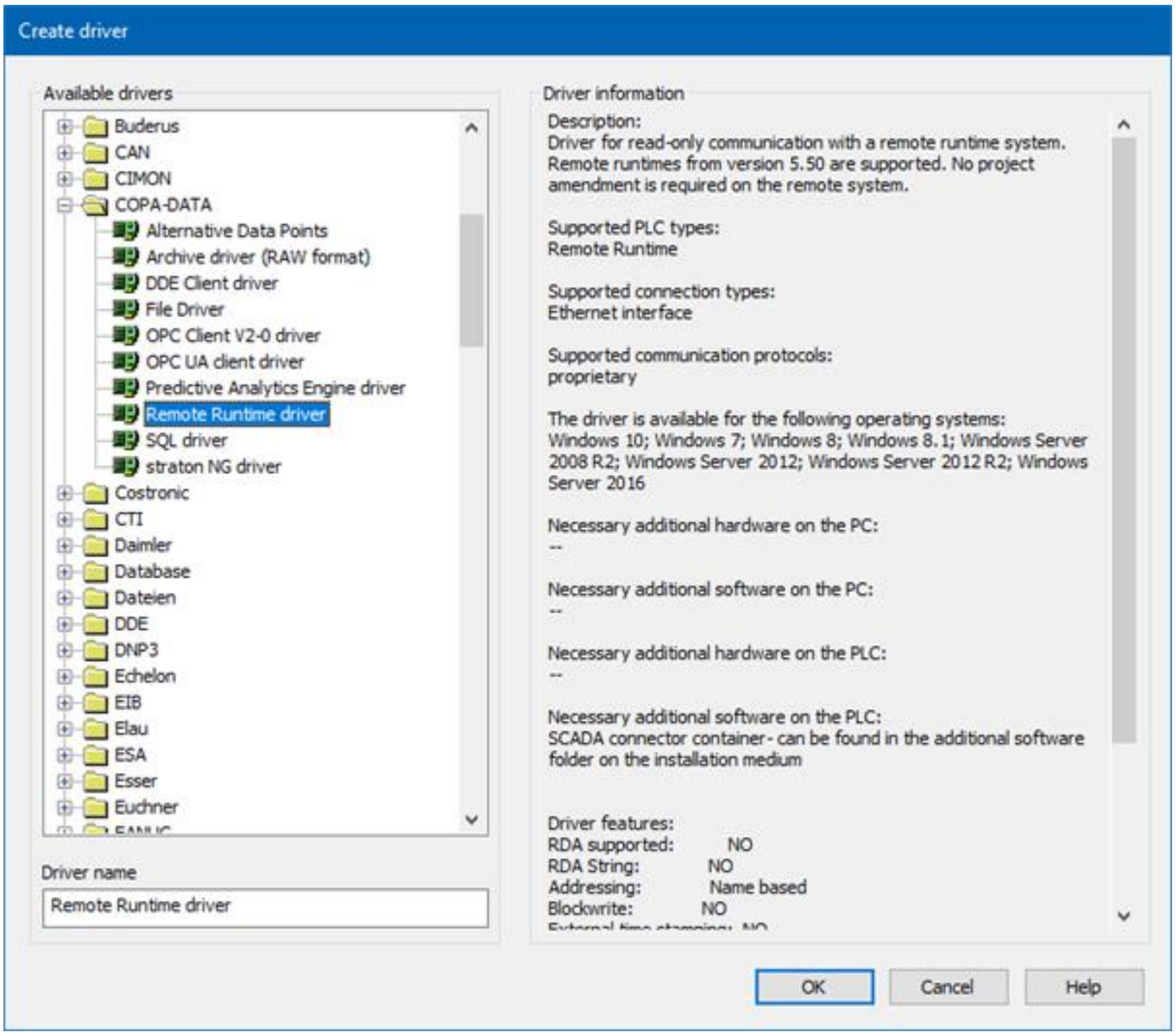
In this chapter you will learn how to use the driver in a project and which settings you can change.

Information

Find out more about further settings for zenon variables in the chapter Variables of the online manual.

6.1 Creating a driver

In the **Create driver** dialog, you create a list of the new drivers that you want to create.



Available drivers:

- Buderus
- CAN
- CIMON
- COPA-DATA
  - Alternative Data Points
  - Archive driver (RAW format)
  - DDE Client driver
  - File Driver
  - OPC Client V2-0 driver
  - OPC UA client driver
  - Predictive Analytics Engine driver
  - Remote Runtime driver
  - SQL driver
  - straton NG driver
- Costronic
- CTI
- Daimler
- Database
- Dateien
- DDE
- DNP3
- Echelon
- EIB
- Elau
- ESA
- Esser
- Euchner

Driver name: Remote Runtime driver

Description:
Driver for read-only communication with a remote runtime system. Remote runtimes from version 5.50 are supported. No project amendment is required on the remote system.

Supported PLC types:
Remote Runtime

Supported connection types:
Ethernet interface

Supported communication protocols:
proprietary

The driver is available for the following operating systems:
Windows 10; Windows 7; Windows 8; Windows 8.1; Windows Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

Necessary additional hardware on the PC:
--

Necessary additional software on the PC:
--

Necessary additional hardware on the PLC:
--

Necessary additional software on the PLC:
SCADA connector container - can be found in the additional software folder on the installation medium

Driver features:
RDA supported: NO
RDA String: NO
Addressing: Name based
Blockwrite: NO

Parameter	Description
Available drivers	List of all available drivers. The display is in a tree structure: *[+]* expands the folder structure and shows the drivers contained therein. [-] reduces the folder structure Default: *No selection*
Driver name	Unique **Identification** of the driver. Default: *empty* The input field is pre-filled with the pre-defined

Parameter	Description
	Identification after selecting a driver from the list of available drivers.
Driver information	Further information on the selected driver. Default: *empty* The information on the selected driver is shown in this area after selecting a driver.

CLOSE DIALOG

Option	Description
OK	Accepts all settings and opens the driver configuration dialog of the selected driver.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

Information

The content of this dialog is saved in the file called Treiber_[Language].xml. You can find this file in the following folder: *C:\ProgramData\COPA-DATA\zenon[version number]*.

CREATE NEW DRIVER

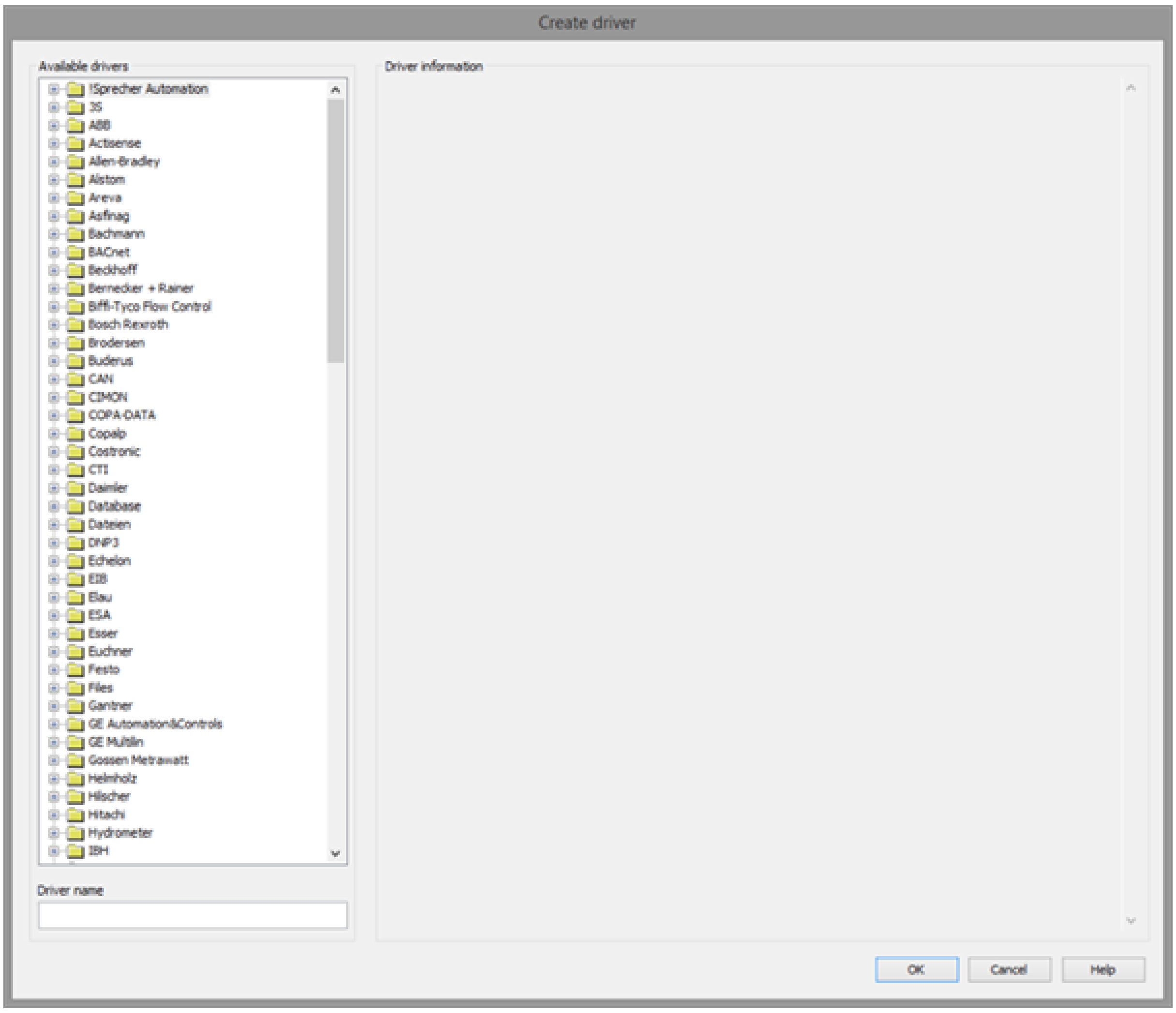
In order to create a new driver:

1. Right-click on **Driver** in the Project Manager and select **New driver** in the context menu.

   Optional: Select the **New driver** button from the toolbar of the detail view of the **Variables**.The Create driver dialog is opened.

   The **Create simple data type** dialog is opened.

2. The dialog offers a list of all available drivers.

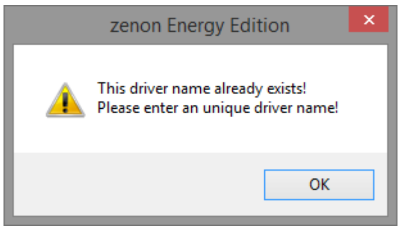


3. Select the desired driver and name it in the **Driver name** input field.
   This input field corresponds to the **Identification** property. The name of the selected driver is automatically inserted into this input field by default.
   The following is applicable for the **Driver name**:
   - The **Driver name** must be unique.
     If a driver is used more than once in a project, a new name has to be given each time.
     This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
   - The **Driver name** is part of the file name.
     Therefore it may only contain characters which are supported by the operating system.
     Invalid characters are replaced by an underscore (_).
   - **Attention:** This name cannot be changed later on.
4. Confirm the dialog by clicking on the **OK** button.
   The configuration dialog for the selected driver is opened.

Note: The language of driver names cannot be switched. They are always shown in the language in which they have been created, regardless of the language of the Editor. This also applies to driver object types.

DRIVER NAME DIALOG ALREADY EXISTS

If there is already a driver in the project, this is shown in a dialog. The warning dialog is closed by clicking on the **OK** button. The driver can be named correctly.


zenon Energy Edition

This driver name already exists!
Please enter an unique driver name!

ZENON PROJECT

The following drivers are created automatically for newly-created projects:

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

Information

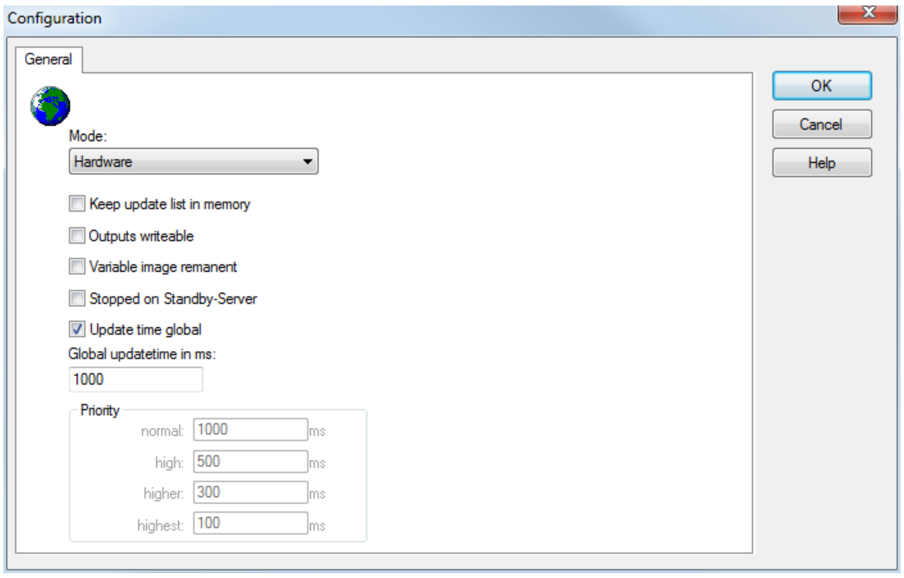
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General

The configuration dialog is opened when a driver is created. In order to be able to open the dialog later for editing, double click on the driver in the list or click on the **Configuration** property.



Mode: Hardware

- [ ] Keep update list in memory
- [ ] Outputs writeable
- [ ] Variable image remanent
- [ ] Stopped on Standby-Server
- [✓] Update time global

Global updatetime in ms: 1000

Priority

normal:	1000	ms
high:	500	ms
higher:	300	ms
highest:	100	ms

Option	Description
Mode	Allows to switch between hardware mode and simulation mode ▸ *Hardware*: A connection to the control is established. ▸ Simulation - static: No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver. ▸ *Simulation - counting*: No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically. ▸ *Simulation - programmed*: No communication is established to the PLC. The

Option	Description
	values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation.
Keep update list in the memory	Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.
Output can be written	- *Active*: Outputs can be written. - *Inactive*: Writing of outputs is prevented. **Note**: Not available for every driver.
Variable image remanent	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp. The variable image is saved in hardware mode if one of these statuses is active: - User status *M1* (*0*) to *M8* (*7*) - *REVISION(9)* - *AUS(20)* - *ERSATZWERT(27)* The variable image is always saved if: - the variable is of the **Communication details** object type - the driver runs in simulation mode. (not programmed simulation) The following states are not restored at the start of the Runtime:

Option	Description
	▸ *SELECT(8)* ▸ *WR-ACK(40)* ▸ *WR-SUC(41)* The mode **Simulation - programmed** at the driver start is not a criterion in order to restore the remanent variable image.
Stop on Standby Server	Setting for redundancy at drivers which allow only one communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade. **Attention:** If this option is active, the gapless archiving is no longer guaranteed. ▸ *Active*: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status **switched off** but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian. Default: *inactive* **Note:** Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.
Global Update time	Setting for the global update times in milliseconds: ▸ *Active*: The set **Global update time** is used for all variables in the project. The priority set at the variables is not used. ▸ *Inactive*: The set priorities are used for the individual variables. **Exceptions:** Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the **Spontaneous driver update time** section.

Option	Description
Priority	The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time. The variables are allocated separately in the settings of the variable properties. The communication of the individual variables can be graded according to importance or required topicality using the priority classes. Thus the communication load is distributed better. **Attention:** Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.

CLOSE DIALOG

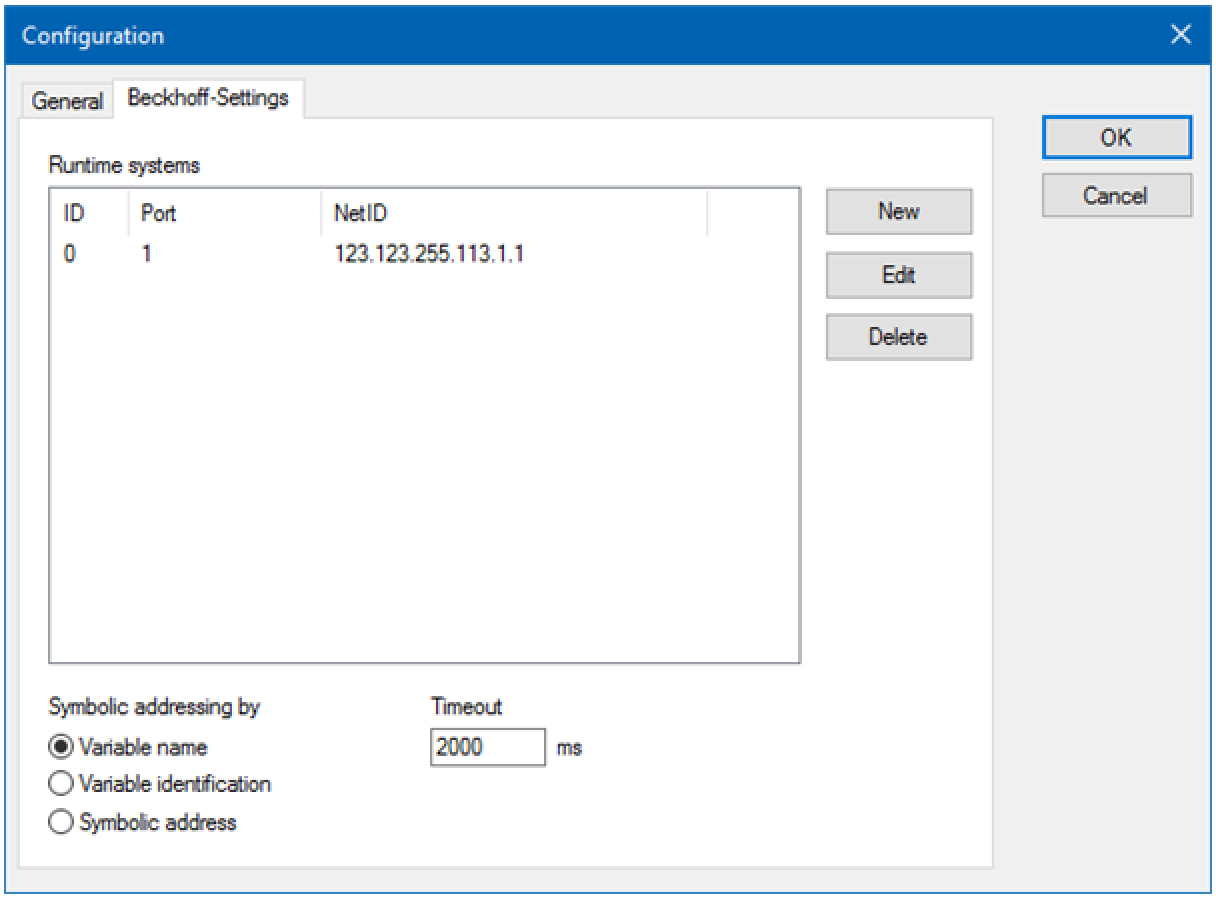
Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

UPDATE TIME FOR SPONTANEOUS DRIVERS

With spontaneous drivers, for **Set value**, **advising** of variables and **Requests**, a read cycle is triggered immediately - regardless of the set update time. This ensures that the value is immediately available for visualization after writing. The update time is generally 100 ms.

Spontaneous drivers are **ArchDrv**, **BiffiDCM**, **BrTcp32**, **DNP3**, **Esser32**, **FipDrv32**, **FpcDrv32**, **IEC850**, **IEC870**, **IEC870_103**, **Otis**, **RTK9000**, **S7DCOS**, **SAIA_Slave**, **STRATON32** and **Trend32**.

6.2.2 Beckhoff settings



Runtime systems

ID	Port	NetID
0	1	123.123.255.113.1.1

Symbolic addressing by

[✓ Variable name]

Variable identification

Symbolic address

Timeout 2000 ms

Parameter	Description
Runtime systems	This list displays all defined runtime systems (**ID**, **Port** and **NetID**). In the variables the defined runtime systems are addressed via the ID. For this the **ID** is entered as **Net address** of the according variable. These Runtime systems have to be notified to the operating system with the TwinCAT software **AMS-ROUTER** or **TcAmsRemoteMgr.exe**. The settings can be changed with the buttons **New**, **Edit** and **Delete**.
Symbolic addressing via	Defines from where the variable addressing via symbol name receives its symbol information. Selection by means of radio buttons: ▸ **Variable name** : Symbol information comes from the **Name**

Parameter	Description
	property of the variables - **Variable identification**: Symbol information comes from the **Identification** property of the zenon variables. To do this, the symbol name must have the same prefix as the variable name. E.g.: *S0_.Symbol name* where *S0_* stands for **Station number** *0* . - **Symbolic address**: Symbol information comes from the **Symbolic address** property of the variables. Default: **Variable name** **Note:** This setting is also considered for the online import. For details see **Addressing** (on page 25) chapter.
Timeout	Entry of the AMS timeout time in milliseconds. The timeout can be set to *2000 ms* for AMS routers. Default: *2000* If there is an error during the communication with the TwinCAT PLC (e.g. the PLC does not respond), the communication will be interrupted after that time and the driver status will be set to *invalid* status. Refer to chapter **Requirements** (on page 8).
New	Opens the dialog to create a new runtime system.
Edit	Opens the dialog to edit the selected runtime system.
Delete	Deletes the selected runtime system from the list.

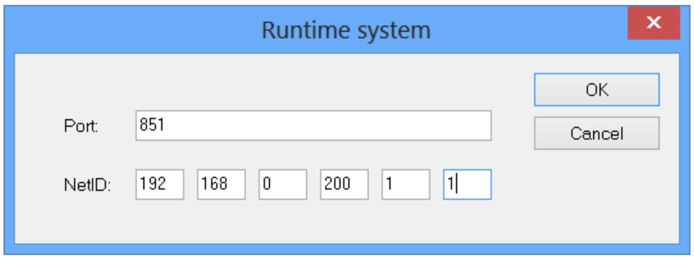
CLOSE DIALOG

Option	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

CREATE/EDIT RUNTIME SYSTEM

To create a runtime system:

1. In the **Beckhoff settings** tab, click on the **New** button.
   The dialog for configuration is opened.
2. Enter **Port** and **NetID**.



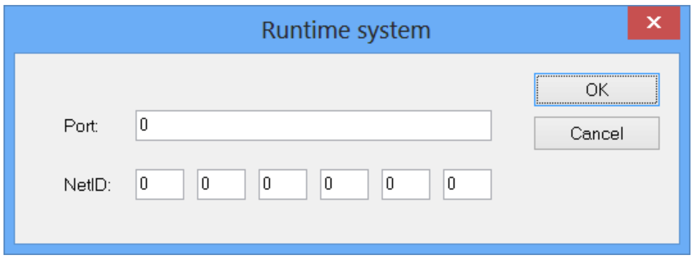
Port: 851

NetID: 192 168 0 200 1 1

To edit an entry:

1. Select the desired runtime system in the list.
2. Click on the **Edit** button.
   The dialog for configuration is opened.
3. Edit the **Port** and **NetID**.

RUNTIME SYSTEM DIALOG



Port: 0

NetID: 0 0 0 0 0 0

Parameter	Description
Port	Entry of the port.
NetID	Entry of the net address with ID.
OK	Applies settings and closes the dialog.
Cancel	Discards settings and closes the dialog.

Information

Standard ADS-Ports:

- TwinCAT 2: *801*
- TwinCAT 3: *851*
- Bus controller (BC9000 for example): *800*

7 Creating variables

This is how you can create variables in the zenon Editor:

7.1 Creating variables in the Editor

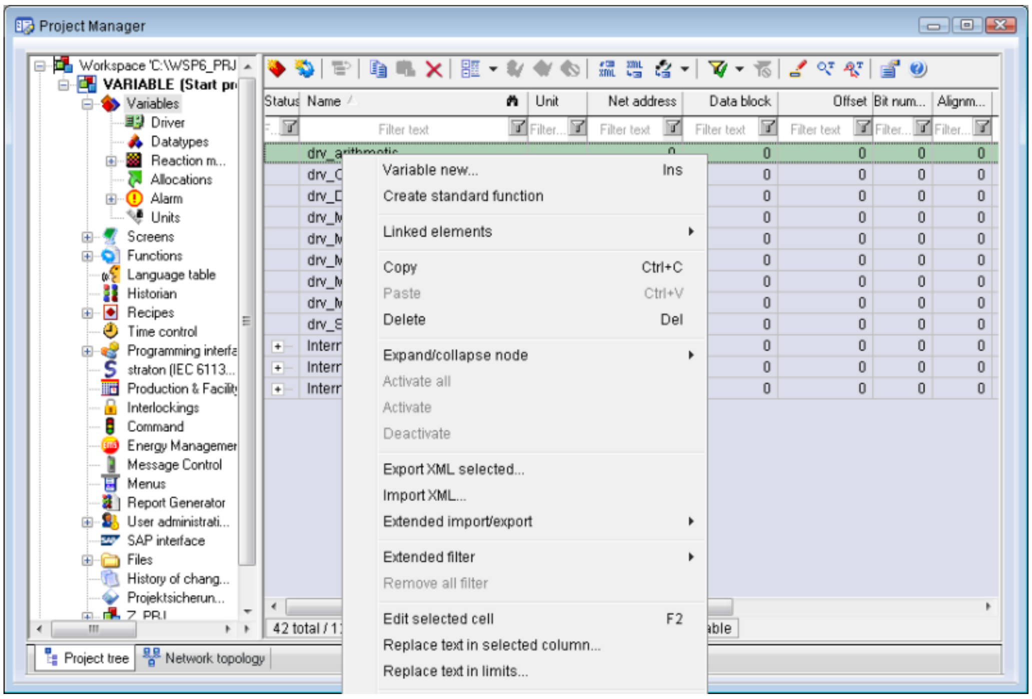
Variables can be created:

- as simple variables
- in arrays
- as structure variables

VARIABLE DIALOG

To create a new variable, regardless of which type:

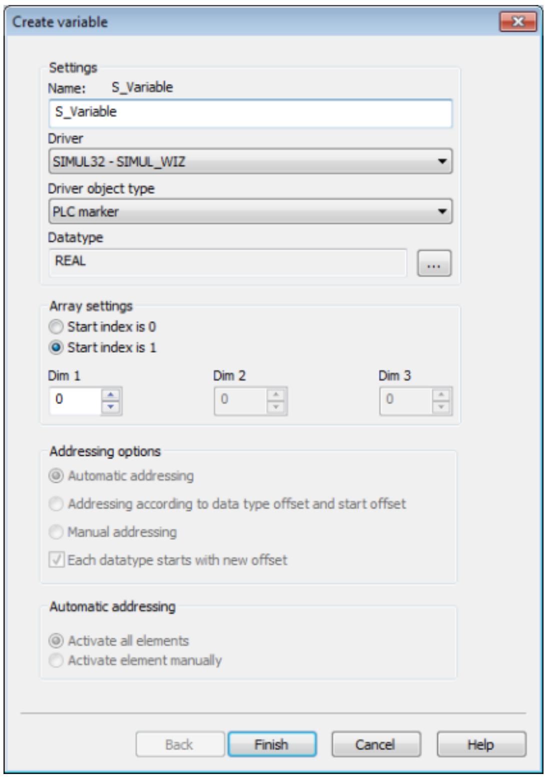
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

2. Configure the variable
3. The settings that are possible depend on the type of variables

CREATE VARIABLE DIALOG


Create variable

Settings

Name: S_Variable

S_Variable

Driver

SIMUL32 - SIMUL_WIZ

Driver object type

PLC marker

Datatype

REAL

Array settings

( ) Start index is 0
(●) Start index is 1

Dim 1: 0 | Dim 2: 0 | Dim 3: 0

Addressing options

(●) Automatic addressing
( ) Addressing according to data type offset and start offset
( ) Manual addressing
[✓] Each datatype starts with new offset

Automatic addressing

(●) Activate all elements
( ) Activate element manually

Property	Description
Name	Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name. Maximum length: 128 characters **Attention:** the characters **#** and **@** are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive. **Note:** Some drivers also allow addressing using the **Symbolic address** property.
Driver	Select the desired driver from the drop-down list. **Note:** If no driver has been opened in the project, the driver for internal variables (**Intern.exe**) is automatically loaded.
Driver Object Type	Select the appropriate driver object type from the drop-down list.
Data Type	Select the desired data type. Click on the ... button to open the selection dialog.
Array settings	Expanded settings for array variables. You can find details in the

Property	Description
	Arrays chapter.
Addressing options	Expanded settings for arrays and structure variables. You can find details in the respective section.
Automatic element activation	Expanded settings for arrays and structure variables. You can find details in the respective section.

SYMBOLIC ADDRESS

The **Symbolic address** property can be used for addressing as an alternative to the **Name** or **Identification** of the variables. Selection is made in the driver dialog; configuration is carried out in the variable property. When importing variables of supported drivers, the property is entered automatically.

Maximum length: *1024* characters.

The following drivers support the **Symbolic address**:

- 3S_V3
- AzureDrv
- BACnetNG
- IEC850
- KabaDPServer
- OPCUA32
- Phoenix32
- POZYTON
- RemoteRT
- S7TIA
- SEL
- SnmpNg32
- PA_Drv
- EUROMAP63

INHERITANCE FROM DATA TYPE

Measuring range, **Signal range** and **Set value** are always:

- derived from the datatype
- Automatically adapted if the data type is changed

Note for signal range: If a change is made to a data type that does not support the set **signal range**, the **signal range** is amended automatically. For example, for a change from **INT** to **SINT**, the **signal range** is changed to *127*. The amendment is also carried out if the **signal range** was not inherited from the data type. In this case, the **measuring range** must be adapted manually.

7.2 Addressing

ACCESS METHODS

Polling - The driver polls the values from the PLC continuously.

NUMBER OF PLCS

One driver can connect to several PLCs.

THE SETTINGS OF THE VARIABLES ARE DEFINED AS FOLLOWS

Settings in the variable	Field from the XML file
Name	Symbol name with station prefix, for example "**S2_SymName**" (see also the information on addressing using a name) Addressing is symbolic using the name and is automatically set during online import in accordance with the configuration. The second "identification" with the properties probably means the "resources label".
Identification	Symbol name with station prefix, for example **"S2_SymName"** (see also the information on addressing using a name in relation to this) Addressing is symbolic using the identification and is automatically set during online import in accordance with the configuration. The second "identification" with the properties probably means the "resources label".
Net address	ID of the according station from the driver

Settings in the variable	Field from the XML file
	configuration. If no station with the according AmsNetID exists, a new station is automatically created. (also see "Beckhoff settings")
Driver connection/Data Type	The datatype is entered according to IEC.
Driver connection/Driver Object Type	As the "Driver object type" a "PLC marker" is used.
Offset	The offset from the PLC
iGroup	The index group from the PLC. Is automatically set during online import.
BitSize	The BitSize from the PLC. Is automatically set during online import. The following bit size must be set for the respective data types:

8	BOOL
8	BYTE
32	DATE
32	DINT
32	DT
32	DATE_AND_TIME
32	DWORD
16	INT
64	LREAL
32	REAL
8	SINT
648	STRING
32	TIME
32	TOD

Settings in the variable	Field from the XML file
	32 TIME_OF_DAY 32 UDINT 16 UINT 8 USINT 16 WORD
Resources label	The comment is used as the identification. All line breaks and tabs are removed.
Driver connection/Priority	not used for this driver The driver does not support cyclically-poling communication in priority classes.
Symbolic address	Addressing is symbolic using the symbolic address and is automatically set during online import in accordance with the configuration. The second "identification" with the properties probably means the "resources label".

OVERWRITING EXISTING VARIABLES (EXTENDED MERGING)

If a variable to be imported already exists in zenon, the following procedure applies:

- If the variable is from another driver, it will not be changed.
- If the variable concerned depends on the current driver, the following are overwritten if changes are made:
  - Data type
  - Driver object type
  - Offset
  - iGroup
  - Bit size
  - Identification

VARIABLE ADDRESSING

Addressing in zenon Runtime is carried out using

- Name
- Identification
- Address

In the first stage, it is established whether a station can be communicate using a name or identification. If this communication type is not successful, the driver will read or write the variables of the according station via the address.

Procedure:

- If there is a symbol with the name, a handle comes back, via which communication then takes place.
- If there is no symbol with the name in the PLC (ADS **Error 0x710**), then:
  - It is logged in the Diagnosis Tool that no symbol exists
  - The variable receives the status *INVALID*
  - There is no attempt to communicate via the offset
- Only if the attempt to get a handle is responded to with the ADS **Error 0x701** (service not supported by server) is an attempt to read via the offset made.

NAME

For addressing using a name, the variable name in zenon has to consist of a prefix and the corresponding symbol name.

- Prefix: **Sn_**
  - **n**: Network address, because the same program could be running on different PLCs
  - Example:

    zenon: **S2_SymbolName**

    PLC: **SymbolName**

IDENTIFICATION

When addressing using a variable, the driver attempts to obtain the symbol names from the variable identification. The following must be the case for this to happen:

- In the driver configuration (on page 18), the checkbox **Identification contains symbol name** must be activated
- The symbol name must have the same prefix as for addressing using a **name**

Online import takes this setting into account and writes the symbol names to the identification of the variables instead of the variable name. In this mode, the online import can be a bit slower, as an allocation list of variable name and variable identification is created before the import starts. In addition, the comments are no longer imported, whilst in addressing mode, the comments are written to the identifier using variable names.

ADDRESS

Parameters	Description
Net address	As the net address the ID of the station created in the configuration dialog (port and AMS net address) is offered for selection.
BitSize	Size of the datapoint in bits
iGroup	Index group in the connected station
Offset	The offset within an iGroup

7.3 Driver objects and datatypes

Driver objects are areas available in the PLC, such as markers, data blocks etc. Here you can find out which driver objects are provided by the driver and which IEC data types can be assigned to the respective driver objects.

7.3.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
PLC marker	8	X	X	*WORD, LREAL, BOOL, DWORD, DINT, DATE_AND_TIME, REAL, UDINT, INT, DATE, TOD, UINT, SINT, USINT, TIME, STRING, WSTRING, LINT, ULINT*	

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
Communication details	35	X	X	*BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING*	Variables for the static analysis of the communication; Values are transferred between driver and Runtime (not to the PLC). **Note**: The addressing and the behavior is the same for most zenon drivers. You can find detailed information on this in the Communication details (Driver variables) (on page 42) chapter.

Key:

X: supported

--: not supported

EXAMPLES OF ALL POSSIBLE IEC DATA TYPES

PLC data type	Data types in zenon
BOOL	*BOOL*
BYTE	*USINT*
DATE	*DATE*
DINT	*DINT*
DT	*DATE_AND_TIME*
DATE_AND_TIME	*DATE_AND_TIME*
DWORD	*DWORD*
INT	*INT*

PLC data type	Data types in zenon
INT16	*INT*
LINT	*LINT*
LREAL	*LREAL*
REAL	*REAL*
SINT	*SINT*
STRING	*STRING*
TIME	*TIME*
TOD	*TOD*
TIME_OF_DAY	*TOD*
UDINT	*UDINT*
UINT	*UINT*
UINT16	*UINT*
ULINT	*ULINT*
USINT	*USINT*
WORD	*WORD*

CHANNEL TYPE

The term **Kanaltyp** is the internal numerical name of the driver object type. It is also used for the extended DBF import/export of the variables.

"**Kanaltyp**" is used for advanced CSV import/export of variables in the "**HWObjectType**" column.

7.3.2 Mapping of the data types

All variables in zenon are derived from IEC data types. The following table compares the IEC datatypes with the datatypes of the PLC.

EXAMPLES OF ALL POSSIBLE ZENON DATA TYPES

PLC	zenon
IX 1.0	Input bit offset 1 bit 1
I 1	Input byte offset 1 low-order
MB 100	Byte marker offset 100 low-order
MX 100.0	Bit marker offset 100 bit 0
MW 100	Word marker offset 100
Q 1	Output byte offset 1 low-order
QW 60000	Output word offset 60000
QX 1.1	Output bit offset 1 bit 1

DRIVER OBJECT TYPES AND SUPPORTED IEC DATA TYPES FOR PROCESS VARIABLES IN ZENON

Driver object types	Channel type	Supported data types (DataType)	Read	Write	Comment
Marker	8	*BOOL* *BYTE* *DATE* *DINT* *DT* *DATE_AND_TIME* *DWORD* *INT* *INT16* *LINT* *LREAL* *REAL* *SINT*	Y	Y	

Driver object types	Channel type	Supported data types (DataType)	Read	Write	Comment
		STRING *TIME* *TOD* *TIME_OF_DAY* *UDINT* *UINT* *UINT16* *ULINT* *USINT* *WORD*			

DATA TYPE

The term **data type** is the internal numerical identification of the data type. It is also used for the extended DBF import/export of the variables.

7.4 Creating variables by importing

Variables can also be imported by importing them. The XML and DBF import is available for every driver.

Information

You can find details on the import and export of variables in the Import-Export manual in the Variables section.

7.4.1 XML import

During XML import of variables or data types, these are first assigned to a driver and then analyzed. Before import, the user decides whether and how the respective element (variable or data type) is to be imported:

- *Import*:
  The element is imported as a new element.
- *Overwrite*:
  The element is imported and overwrites a pre-existing element.
- *Do not import*:
  The element is not imported.

Note: The actions and their durations are shown in a progress bar during import. The import of variables is described in the following documentation. Data types are imported along the same lines.

REQUIREMENTS

The following conditions are applicable during import:

- **Backward compatibility**

  At the XML import/export there is no backward compatibility. Data from older zenon versions can be taken over. The handover of data from newer to older versions is not supported.

- **Consistency**

  The XML file to be imported has to be consistent. There is no plausibility check on importing the file. If there are errors in the import file, this can lead to undesirable effects in the project.

  Particular attention must be paid to this, primarily if not all properties exist in the XML file and these are then filled with default values. E.g.: A binary variable has a limit value of *300*.

- **Structure data types**

  Structure data types must have the same number of structure elements.
  Example: A structure data type in the project has 3 structure elements. A data type with the same name in the XML file has 4 structure elements. Then none of the variables based on this data type in the file are imported into the project.

Hint

You can find further information on XML import in the **Import - Export** manual, in the **XML import** chapter.

7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.

Information

Import and Export via CSV or dBase supported; no driver specific variable settings, such as formulas. Use export/import via XML for this.

IMPORT DBF FILE

To start the import:

1. right-click on the variable list.
2. In the drop-down list of **Extended export/import...** select the **Import dBase** command.
3. Follow the instructions of the import assistant.

The format of the file is described in the chapter File structure.

Information

Note:

- Driver object type and data type must be amended to the target driver in the DBF file in order for variables to be imported.
- dBase does not support structures or arrays (complex variables) at import.

EXPORT DBF FILE

To start the export:

1. right-click on the variable list.
2. In the drop-down list of **Extended export/import...** select the **Export dBase...** command .
3. Follow the instructions of the import assistant.

Attention

DBF files:

- must correspond to the 8.3 DOS format for filenames (8 alphanumeric characters for name, 3 character suffix, no spaces)
- must not have dots (.) in the path name.
  e.g. the path *C:\users\John.Smith\test.dbf* is invalid.
  Valid: *C:\users\JohnSmith\test.dbf*
- must be stored close to the root directory in order to fulfill the limit for file name length including path: maximum 255 characters

The format of the file is described in the chapter File structure.

Information

dBase does not support structures or arrays (complex variables) at export.

FILE STRUCTURE OF THE DBASE EXPORT FILE

The dBaseIV file must have the following structure and contents for variable import and export:

Attention

dBase does not support structures or arrays (complex variables) at export.

DBF files must:

- conform with their name to the 8.3 DOS format (8 alphanumeric characters for name, 3 characters for extension, no space)
- Be stored close to the root directory (Root)

STRUCTURE

Identification	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered manually). The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
KANAL_D	Log	1	The variable is deleted with the *1* entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the

Identification	Type	Field size	Comment
			data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Net address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager
LES_SCHR	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTTYP and DATENTYP
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables

Identification	Typ e	Field size	Comment
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix
ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists. The length can be limited using the MAX_LAENGE entry in **project.ini**.
ADJWVBA	R	1	Non-linear value adaption: *0*: Non-linear value adaption is used *1*: Non-linear value adaption is not used
ADJZENON	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
ZWREMA	N	16	Linked counter REMA.

Identification	Typ e	Field size	Comment
MAXGRAD	N	16	Gradient overflow for counter REMA.

Attention

When importing, the driver object type and data type must be amended to the target driver in the DBF file in order for variables to be imported.

LIMIT VALUE DEFINITION

Limit definition for limit values *1* to *4*, or status *1* to *4*:

Identification	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	technical value or ID number of a linked variable for a dynamic limit value (see VARIABLEx) (if VARIABLEx is *1* and here it is *-1*, the existing variable linkage is not overwritten)
SCHWWERT1	F	16	Threshold value for limit value
HYSTERESE1	F	14	Is not used
BLINKEN1	R	1	Set blink attribute
BTB1	R	1	Logging in CEL
ALARM1	R	1	Alarm
DRUCKEN1	R	1	Printer output (for CEL or Alarm)
QUITTIER1	R	1	Must be acknowledged
LOESCHE1	R	1	Must be deleted
VARIABLE1	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	R	1	Functions linking
ASK_FUNC1	R	1	Execution via Alarm Message List
FUNC_NR1	N	10	ID number of the linked function

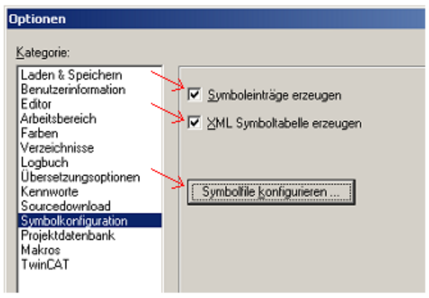
Identification	Type	Field size	Comment
			(if "-1" is entered here, the existing function is not overwritten during import)
A_GRUPPE1	N	10	Alarm/Event Group
A_KLASSE1	N	10	Alarm/Event Class
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Color as Windows coding
GRENZTXT1	C	66	Limit value text
A_DELAY1	N	10	Time delay
INVISIBLE1	R	1	Invisible

Expressions in the column "Comment" refer to the expressions used in the dialog boxes for the definition of variables. For more information, see chapter Variable definition.

7.4.3 Online import

IMPORT VARIABLES ONLINE

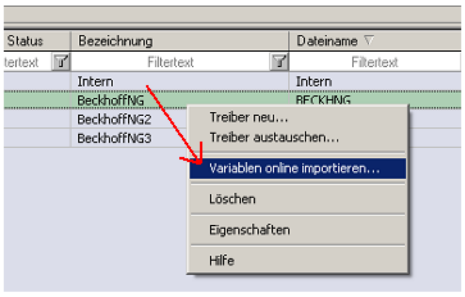
Variables can be imported into zenon via the symbol file (*.tpy). For this, you have make sure to activate the checkboxes **Create symbol entries** and **Create XML symbol table** under Options/Symbol configuration in the according TwinCAT project. The symbol file also has to be configured accordingly (see button below).



✓ Symboleinträge erzeugen

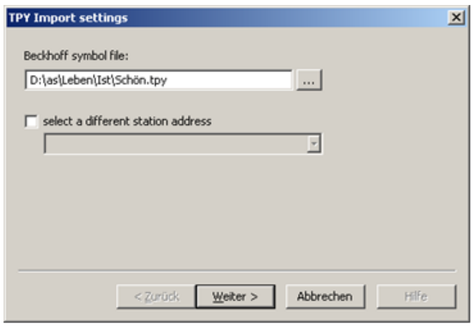
✓ XML Symboltabelle erzeugen

In zenon, the variables can be easily created and changed with the online import. To do this, right-click on the desired driver in the zenon Editor and select the command **Import variables online...** from the context menu.



Status	Bezeichnung	Dateiname
tertext	Filtertext	Filtertext
	Intern	Intern
	BeckhoffNG	BECKHNG
	BeckhoffNG2	
	BeckhoffNG3	

You will then be asked for the path of the symbol file. After confirming the selection with **Finish**, the symbol file will be loaded. This can take several seconds depending on the number of variables. In this dialog you can select a station already created in the configuration, for which the variables should be imported. If no station is selected here, the AMS NET-ID from the .TPY file will be used, and if this station does not exist, it will be created.

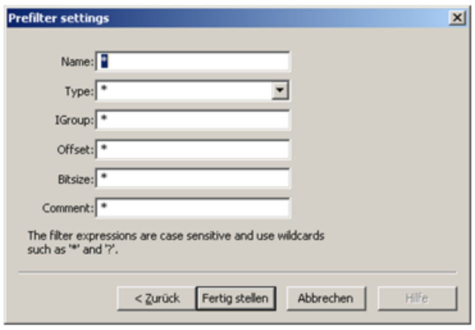


TPY Import settings

Beckhoff symbol file: D:\as\Leben\Ist\Schön.tpy

☐ select a different station address

Attention: If there is no AMS net ID in the .tpy file, a station with the AMS net ID "0.0.0.0.0.0" is created automatically. During import, manually select the correct AMS net ID or amend the AMS Net ID in the driver configuration. A .tpy file from TwinCAT 3.x does not contain an AMS net ID.



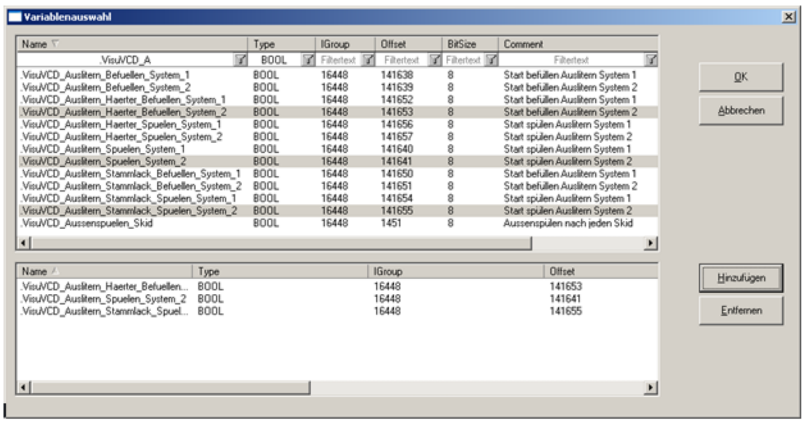
Prefilter settings

Name:	*
Type:	*
IGroup:	*
Offset:	*
Bitsize:	*
Comment:	*

The filter expressions are case sensitive and use wildcards such as '*' and '?'.

After that you can define a prefilter. Here you can e.g. filter by comments in the TwinCAT program, so that only certain variables will be available for the visualization. Using a prefilter particularly makes sense for large TwinCAT projects with many structures.

As a result, a selection dialog with a flat (structures and arrays are resolved) symbol list is opened. Use the column headers for filtering and sorting. You can change the selection of the variables to be imported using the **Add** and **Remove** buttons.



Name	Type	IGroup	Offset	BitSize	Comment
.VisuVCD_A	BOOL	Filtertext	Filtertext	Filtertext	Filtertext
.VisuVCD_Auslitern_Befuellen_System_1	BOOL	16448	141638	8	Start befüllen Auslitern System 1
.VisuVCD_Auslitern_Befuellen_System_2	BOOL	16448	141639	8	Start befüllen Auslitern System 2
.VisuVCD_Auslitern_Haerter_Befuellen_System_1	BOOL	16448	141652	8	Start befüllen Auslitern System 1
.VisuVCD_Auslitern_Haerter_Befuellen_System_2	BOOL	16448	141653	8	Start befüllen Auslitern System 2
.VisuVCD_Auslitern_Haerter_Spuelen_System_1	BOOL	16448	141656	8	Start spülen Auslitern System 1
.VisuVCD_Auslitern_Haerter_Spuelen_System_2	BOOL	16448	141657	8	Start spülen Auslitern System 2
.VisuVCD_Auslitern_Spuelen_System_1	BOOL	16448	141640	8	Start spülen Auslitern System 1
.VisuVCD_Auslitern_Spuelen_System_2	BOOL	16448	141641	8	Start spülen Auslitern System 2
.VisuVCD_Auslitern_Stammlack_Befuellen_System_1	BOOL	16448	141650	8	Start befüllen Auslitern System 1
.VisuVCD_Auslitern_Stammlack_Befuellen_System_2	BOOL	16448	141651	8	Start befüllen Auslitern System 2
.VisuVCD_Auslitern_Stammlack_Spuelen_System_1	BOOL	16448	141654	8	Start spülen Auslitern System 1
.VisuVCD_Auslitern_Stammlack_Spuelen_System_2	BOOL	16448	141655	8	Start spülen Auslitern System 2
.VisuVCD_Aussenspuelen_Skid	BOOL	16448	1451	8	Aussenspülen nach jeden Skid

Name	Type	IGroup	Offset
.VisuVCD_Auslitern_Haerter_Befuellen...	BOOL	16448	141653
.VisuVCD_Auslitern_Spuelen_System_2	BOOL	16448	141641
.VisuVCD_Auslitern_Stammlack_Spuel...	BOOL	16448	141655

After closing the dialog with **Ok**, the selected variables will be imported in zenon.

Information

The online import regards the setting **Identification includes symbol name** from Driver dialog Beckhoff settings (on page 18):

Active: The symbol name is written to the identification of the variable instead of the variable name. In this mode the online import can be a bit slower as a allocation list of variable name and variable identification is created before the import starts. Comments are not imported.

Inactive: The addressing takes place via the variable name. The comment is written to the identification.

7.5 Communication details (Driver variables)

The driver kit implements a number of driver variables. This variables are part of the driver object type *Communication details*. These are divided into:

- Information
- Configuration
- Statistics and
- Error message

The definitions of the variables implemented in the driver kit are available in the import file **DRVVAR.DBF** and can be imported from there.
Path to file: *%ProgramData%\COPA-DATA\zenon<Versionsnummer>\PredefinedVariables*

Note: Variable names must be unique in zenon. If driver variables of the driver object type *Communication details* are to be imported from **DRVVAR.DBF** again, the variables that were imported beforehand must be renamed.

Information

Not every driver supports all driver variables of the driver object type *Communication details*.

For example:

- Variables for modem information are only supported by modem-compatible drivers.
- Driver variables for the polling cycle are only available for pure polling drivers.
- Connection-related information such as **ErrorMSG** is only supported for drivers that only edit one connection at a a time.

INFORMATION

Name from import	Type	Offset	Description
MainVersion	*UINT*	0	Main version number of the driver.
SubVersion	*UINT*	1	Sub version number of the driver.
BuildVersion	*UINT*	29	Build version number of the driver.
RTMajor	*UINT*	49	zenon main version number
RTMinor	*UINT*	50	zenon sub version number
RTSp	*UINT*	51	zenon Service Pack number
RTBuild	*UINT*	52	zenon build number
LineStateIdle	*BOOL*	24.0	TRUE, if the modem connection is idle
LineStateOffering	*BOOL*	24.1	TRUE, if a call is received
LineStateAccepted	*BOOL*	24.2	The call is accepted
LineStateDialtone	*BOOL*	24.3	Dialtone recognized
LineStateDialing	*BOOL*	24.4	Dialing active
LineStateRingBack	*BOOL*	24.5	While establishing the connection
LineStateBusy	*BOOL*	24.6	Target station is busy

Name from import	Type	Offset	Description
LineStateSpecialInfo	*BOOL*	24.7	Special status information received
LineStateConnected	*BOOL*	24.8	Connection established
LineStateProceeding	*BOOL*	24.9	Dialing completed
LineStateOnHold	*BOOL*	24.10	Connection in hold
LineStateConferenced	*BOOL*	24.11	Connection in conference mode.
LineStateOnHoldPendConf	*BOOL*	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	*BOOL*	24.13	Connection in hold for transfer
LineStateDisconnected	*BOOL*	24.14	Connection terminated.
LineStateUnknow	*BOOL*	24.15	Connection status unknown
ModemStatus	*UDINT*	24	Current modem status
TreiberStop	*BOOL*	28	Driver stopped For *driver stop*, the variable has the value *TRUE* and an **OFF** bit. After the driver has started, the variable has the value *FALSE* and no **OFF** bit.
SimulRTState	*UDINT*	60	Informs the state of Runtime for driver simulation.
ConnectionStates	*STRING*	61	Internal connection status of the driver to the PLC. Connection statuses: ▶ *0:* Connection OK ▶ *1:* Connection failure ▶ *2:* Connection simulated Formating: **<Net address>:<Connection status>;…;…;** A connection is only known after a variable has first signed in. In order for a connection to be contained in a string, a variable of this connection must be signed in once.

Name from import	Type	Offset	Description
			The status of a connection is only updated if a variable of the connection is signed in. Otherwise there is no communication with the corresponding controller.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	*BOOL*	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	*BOOL*	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).
ApplyModem	*BOOL*	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings **PhoneNumberSet** and **ModemHwAdrSet**.
PhoneNumberSet	*STRING*	38	Telephone number, that should be used
ModemHwAdrSet	*DINT*	39	Hardware address for the telephone number
GlobalUpdate	*UDINT*	3	Update time in milliseconds (ms).
BGlobalUpdaten	*BOOL*	4	TRUE, if update time is global
TreiberSimul	*BOOL*	5	TRUE, if driver in sin simulation mode
TreiberProzab	*BOOL*	6	TRUE, if the variables update list should be kept in the memory
ModemActive	*BOOL*	7	TRUE, if the modem is active for the driver
Device	*STRING*	8	Name of the serial interface or name of the modem
ComPort	*UINT*	9	Number of the serial interface.
Baudrate	*UDINT*	10	Baud rate of the serial interface.

Name from import	Type	Offset	Description
Parity	*SINT*	11	Parity of the serial interface
ByteSize	*USINT*	14	Number of bits per character of the serial interface Value = *0* if the driver cannot establish any serial connection.
StopBit	*USINT*	13	Number of stop bits of the serial interface.
Autoconnect	*BOOL*	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	*STRING*	17	Current telephone number
ModemHwAdr	*DINT*	21	Hardware address of current telephone number
RxIdleTime	*UINT*	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)
WriteTimeout	*UDINT*	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	*UDINT*	20	Number of ringing tones before a call is accepted
ReCallIdleTime	*UINT*	53	Waiting time between calls in seconds (s).
ConnectTimeout	*UINT*	54	Time in seconds (s) to establish a connection.

STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	*UDINT*	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	*UDINT*	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	*UDINT*	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	*UDINT*	41	Shortest time in milliseconds (ms) that is required to read a data block.

Name from import	Type	Offset	Description
WriteErrorCount	*UDINT*	33	Number of writing errors
ReadSucceedCount	*UDINT*	35	Number of successful reading attempts
MaxCycleTime	*UDINT*	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	*UDINT*	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	*UDINT*	26	Number of writing attempts
ReadErrorCount	*UDINT*	34	Number of reading errors
MaxUpdateTimeNormal	*UDINT*	56	Time since the last update of the priority group **Normal** in milliseconds (ms).
MaxUpdateTimeHigher	*UDINT*	57	Time since the last update of the priority group **Higher** in milliseconds (ms).
MaxUpdateTimeHigh	*UDINT*	58	Time since the last update of the priority group **High** in milliseconds (ms).
MaxUpdateTimeHighest	*UDINT*	59	Time since the last update of the priority group **Highest** in milliseconds (ms).
PokeFinish	*BOOL*	55	Goes to *1* for a query, if all current pokes were executed

ERROR MESSAGE

Name from import	Type	Offset	Description
ErrorTimeDW	*UDINT*	2	Time (in seconds since 1.1.1970), when the last error occurred.
ErrorTimeS	*STRING*	2	Time (in seconds since 1.1.1970), when the last error occurred.
RdErrPrimObj	*UDINT*	42	Number of the PrimObject, when the last reading error occurred.
RdErrStationsName	*STRING*	43	Name of the station, when the last reading error occurred.
RdErrBlockCount	*UINT*	44	Number of blocks to read when the last reading error occurred.

Name from import	Type	Offset	Description
RdErrHwAdresse	*DINT*	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	*UDINT*	46	Block number when the last reading error occurred.
RdErrMarkerNo	*UDINT*	47	Marker number when the last reading error occurred.
RdErrSize	*UDINT*	48	Block size when the last reading error occurred.
DrvError	*USINT*	25	Error message as number
DrvErrorMsg	*STRING*	30	Error message as text
ErrorFile	*STRING*	15	Name of error log file

8 Driver-specific functions

The driver supports the following functions:

LIMITATIONS

When connecting to a Beckhoff Buscontroller (BC9000 for example), communication is first attempted by means of a handle. As a bus controller of this type does not support this type of communication, communication will be delayed. Instead of the **BeckhNG** driver, the **BeckhBC** driver should be used here. This driver has been specially developed for this type of controller.

BLOCK ARRAYS

- Block arrays can be read.
  - Name-based controllers:
    In doing so, the symbol name is used without indexes.
    Example: **MAIN.MyArray**. Square brackets cannot be used.
  - Address-based controllers:
    In doing so, the address of the first index is used.
- The number of indexes must correspond to that in the controller.
  Example:
  For **MyArray: ARRAY [0..99] OF INT;**

- in the zenon variable properties, in the **Additional settings** group, the value of the **Block array size** property must be set to *100*

BLOCKWRITE

The driver does not support blockwrite.

REAL TIME STAMPING

The driver does not support real-time time stamping.

ERROR FILE

Errors are written to the central zenon LOG and can be evaluated with the Diagnosis Viewer.

EXTENDED ERROR FILE

The driver does not write an extended error file.

REDUNDANCY

Redundant operation is possible with this driver.

RDA

The **BeckhNG** driver uses a property of the ADS interface that makes it possible to read the whole binary data block of variables with complex types (structures, arrays) as a whole, using the base name.

The driver determines the base name by separating the last dot (.) from the symbol name of the trigger variable and using the name it has obtained this way. The trigger variable must be the first element of a structure in order for this method to work. The rest of the structure can be created as desired. When reading the overall structure, only the binary data that has been read in when creating the RDA data according to the zenon   documentation needs to correspond.

EXAMPLES

RDA-TYP 3:

```
TYPE RDA_DATA_3 : (* Structure for RDA type 3 payload *)

STRUCT

Value : DINT; (* value

TimeStamp : ARRAY[0..7] OF BYTE; (* Time stamp (year, month, day, hour, minute, second, 1/100th
second, reserve) *)

END_STRUCT
```

```
END_TYPE

TYPE RDA_3 : (* Structure for RDA type 3 *)
STRUCT
Trigger : DINT; (* trigger variable *)
Count : UDINT; (* Number of data sets *)
Cycle : UDINT; (* Cycle time in [ms] (only relevant for type 1 and 4 *)
RDA_Type : UDINT; (* RDA type, 1 - 4 *)
Oldest : UDINT; (* Index of the oldest value (placeholder for compatibility reasons, only relevant for type 1) *)
Data : ARRAY[0..19] OF RDA_DATA_3; (* payload *)
END_STRUCT
END_TYPE
```

RDA-TYP 4:

```
TYPE RDA_4 : (* Structure for RDA type 4 *)
STRUCT
Trigger : DINT; (* trigger variable *)
Count : UDINT; (* Number of data sets *)
Cycle : UDINT; (* Cycle time in [ms] (only relevant for type 1 and 4 *)
RDA_Type : UDINT; (* RDA type, 1 - 4 *)
Oldest : UDINT; (* Index of the oldest value (placeholder for compatibility reasons, only relevant for type 1) *)
TimeStamp : ARRAY[0..7] OF BYTE; (* Time stamp of the first value (year, month, day, hour, minute, second, 1/100th second, reserve)) *)
Data : ARRAY[0..19] OF DINT; (* Payload *)
END_STRUCT
```

END_TYPE

If, for example, an *RDA_4* type **RDA_Test** variable is created in the TwinCAT project, then a variable with the symbolic address **RDA_Test.Trigger** is created in zenon, and *HD active* and *Updated values* are set.

If the variable changes value from *0* to *1*, the **.Trigger** part of the symbol name **RDA_Test.Trigger** is cut off and **RDA_Test** is read in as a binary data block. The RDA processing is then carried out with this data block.

9 Driver command function

The zenon **Driver commands** function is to influence drivers using zenon.
You can do the following with a driver command:

- Start
- Stop
- Shift a certain driver mode
- Instigate certain actions

Note: This chapter describes standard functions that are valid for most zenon drivers.
Not all functions described here are available for every driver. For example, a driver that does not, according to the data sheet, support a modem connection also does not have any modem functions.

Attention

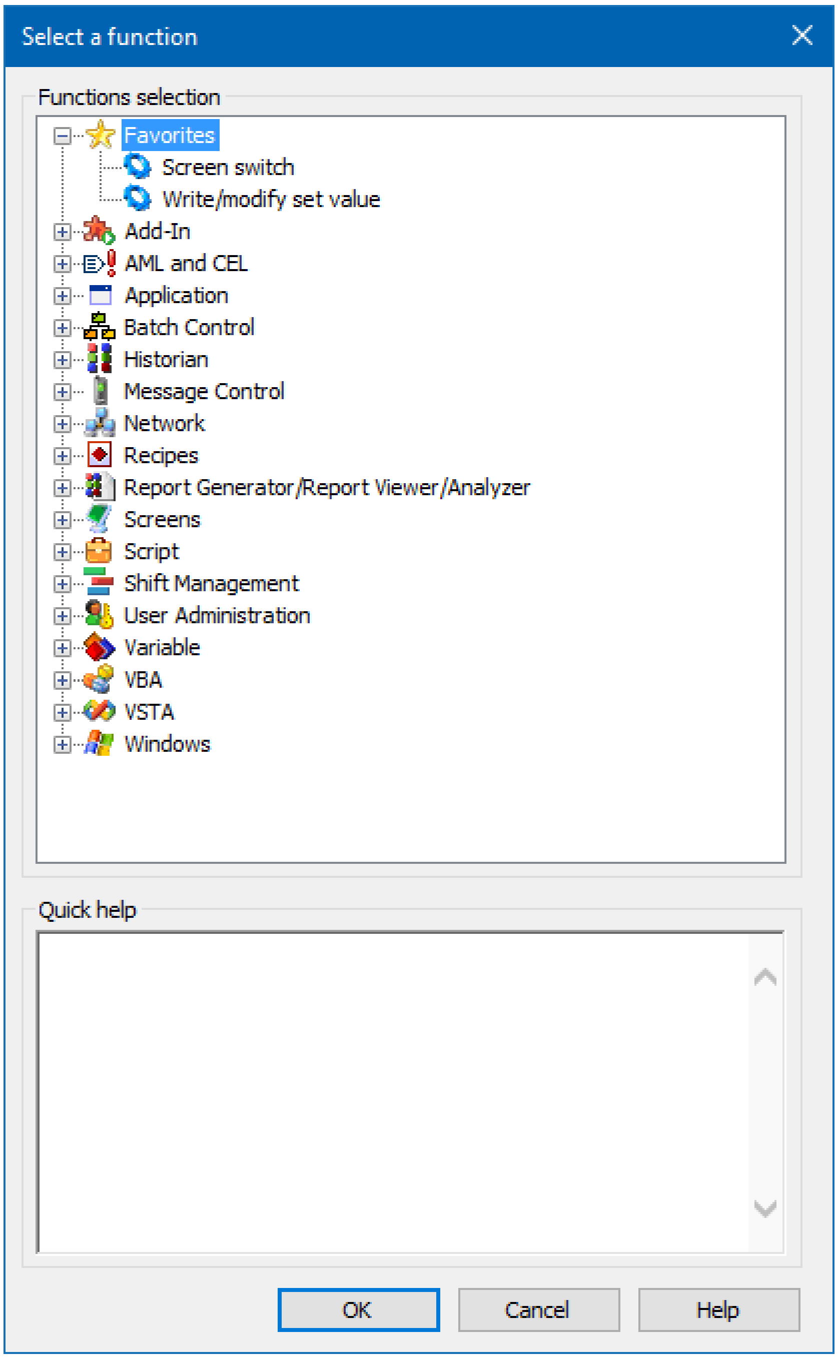
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

Configuration is carried out using the **Driver commands** function.
To configure the function:

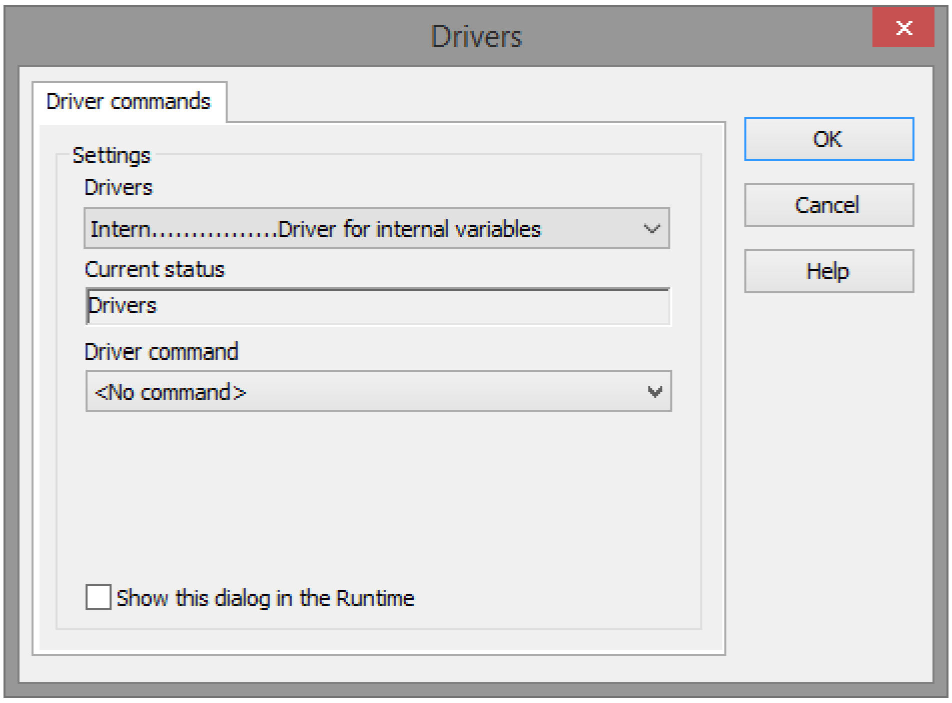
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



2. Navigate to the node **Variable.**
3. Select the **Driver commands** entry.

The dialog for configuration is opened


Drivers

Driver commands

Settings

Drivers: Intern................Driver for internal variables

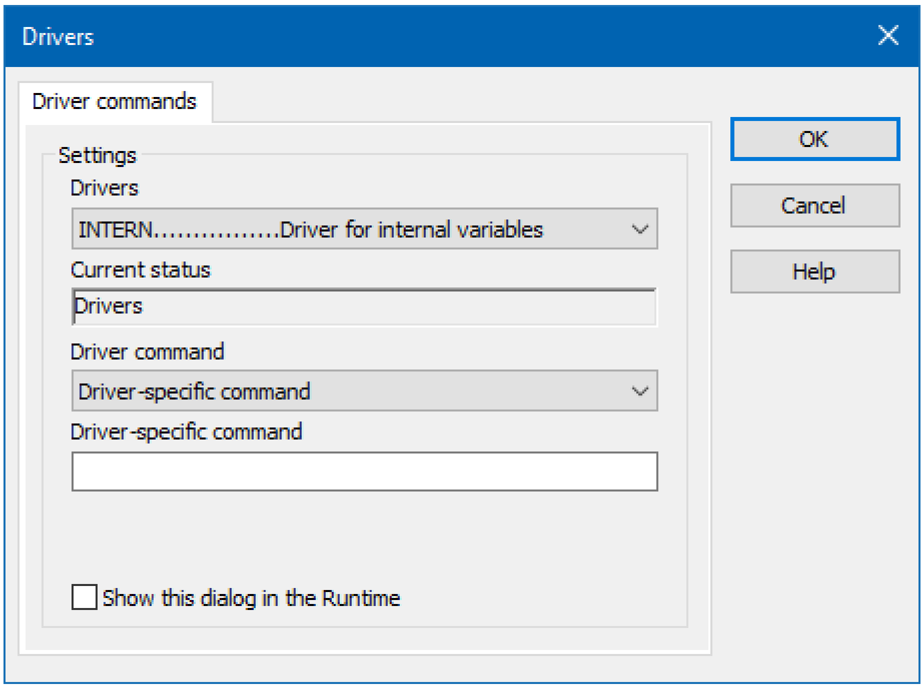
Current status: Drivers

Driver command: <No command>

☐ Show this dialog in the Runtime

4. Select the desired driver and the required command.
5. Close the dialog by clicking on **OK** and ensure that the function is executed in the Runtime. Heed the notices in the **Driver command function in the network** section.

DRIVER COMMAND DIALOG


Drivers

Driver commands

Settings

Drivers: INTERN................Driver for internal variables

Current status: Drivers

Driver command: Driver-specific command

Driver-specific command:

☐ Show this dialog in the Runtime

Option	Description
Driver	Selection of the driver from the drop-down list. It contains all drivers loaded in the project.
Current condition	Fixed entry that is set by the system. no function in the current version.
Driver command	no function in the current version. For details on the configurable driver commands, see the **available driver commands** section.
Driver-specific command	Entry of a command specific to the selected driver. **Note:** Only available if, for the **driver command** option, the *driver-specific command* has been selected.
Show this dialog in the Runtime	Configuration of whether the configuration can be changed in the Runtime: - *Active*: This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution. - *Inactive*: The Editor configuration is applied in the Runtime when executing the function. Default: *inactive*

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

AVAILABLE DRIVER COMMANDS

These driver commands are available - depending on the selected driver:

Driver command	Description
No command	No command is sent. A command that already exists can thus be removed from a configured function.

Driver command	Description
Start driver (online mode)	Driver is reinitialized and started. **Note:** If the driver has already been started, it must be stopped. Only then can the driver be re-initialized and started.
Stop driver (offline mode)	Driver is stopped. No new data is accepted. **Note:** If the driver is in offline mode, all variables that were created for this driver receive the status *switched off* (*OFF*; Bit *20*).
Driver in simulation mode	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
Driver in hardware mode	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
Driver-specific command	Entry of a driver-specific command. Opens input field in order to enter a command.
Driver - activate set setpoint value	Write set value to a driver is possible.
Driver - deactivate set setpoint value	Write set value to a driver is prohibited.
Establish connecton with modem	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
Disconnect from modem	Terminate connection (for modem drivers)
Driver in counting simulation mode	Driver is set into counting simulation mode. All values are initialized with *0* and incremented in the set update time by *1* each time up to the maximum value and then start at *0* again.
Driver in static simulation mode	No communication to the controller is established. All values are initialized with *0*.
Driver in programmed simulation mode	The values are calculated by a freely-programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime.

DRIVER COMMAND FUNCTION IN THE NETWORK

If the computer on which the **Driver commands** function is executed is part of the zenon network, further actions are also carried out:

- A special network command is sent from the computer to the project server.
  It then executes the desired action on its driver.
- In addition, the Server sends the same driver command to the project standby.
  The standby also carries out the action on its driver.

This makes sure that Server and Standby are synchronized. This only works if the Server and the Standby both have a working and independent connection to the hardware.

10 Error analysis

Should there be communication problems, this chapter will assist you in finding out the error.

10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer program that was also installed with zenon. You can find it under **Start/All programs/zenon/Tools 8.20 -> Diagviewer.**

zenon driver log all errors in the LOG files.LOG files are text files with a special structure. The default folder for the LOG files is subfolder **LOG** in the folder **ProgramData**. For example:

%ProgramData%\COPA-DATA\LOG.

Attention: With the default settings, a driver only logs error information. With the Diagnosis Viewer you can enhance the diagnosis level for most of the drivers to "Debug" and "Deep Debug". With this the driver also logs all other important tasks and events.

In the Diagnosis Viewer you can also:

- Follow newly-created entries in real time
- customize the logging settings
- change the folder in which the LOG files are saved

Note:

1. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.

2. The Diagnosis Viewer does not display all columns of a LOG file per default. To display more columns activate property **Add all columns with entry** in the context menu of the column header.
3. If you only use **Error-Logging**, the problem description is in the column **Error text**. For other diagnosis level the description is in the column **General text**.
4. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** or **Error code** or **Driver error parameter** (**1** and **2**). Hints on the meaning of error codes can be found in the driver documentation and the protocol/PLC description.
5. At the end of your test set back the diagnosis level from **Debug** or **Deep Debug**. At **Debug** and **Deep Debug** there are a great deal of data for logging which are saved to the hard drive and which can influence your system performance. They are still logged even after you close the Diagnosis Viewer.

   Attention

   In Windows CE errors are not logged per default due to performance reasons.

You can find further information on the Diagnosis Viewer in the Diagnose Viewer manual.

10.2 Driver monitoring

Runtime monitors the availability of the driver by means of a watchdog. If a driver is no longer available, the *INVALID* status bit is also set for all checked-in variables.

Possible causes for a triggering of the watchdog:

- The driver process is no longer running.

  Check whether the driver EXE file is still running in the Task Manager.
- Operating system is busy with processes that have a higher priority.

  Check the configuration of your system to see whether there is sufficient memory and CPU power. In this case, the driver only resets the *INVALID* status bit if there is a value change on the connected party. Static values retain the *INVALID* status bit until the next time the Runtime or the driver is started.

CONFIGURATION OF WATCHDOG

For the monitoring of communication in the Runtime, the connection to the driver is checked in a fixed, prescribed time period of 60 seconds. This process is repeated several times. If, within 5 attempts (= within 5 minutes), no valid connection to the driver is detected, the *INVALID* bit is set for the checked-in (*advised*) variables. In addition, the *INVALID* bit is also set when new variables are advised. The *INVALID* bit will no longer be reset.

Corresponding LOG entries are created for this.

LOG ENTRY

An error message is logged in the LOG when the watchdog is triggered:

Parameter	Description
Communication with driver:<drvExe>/<drvDesc>(id:<drvId>) timed out. No communication for <time> ms.	No communication with driver within the given time. ▶ *<time>*: Time (in milliseconds) ▶ *<drvDesc>*: Driver name ▶ *<drvExe>*: Driver EXE name ▶ *<drvId>*: Driver ID in the zenon project
Communication with %s timed out. Invalid-Bit will be set.	Communication to the *%s* driver could not be established after 5 attempts within 60 seconds. The *INVALID* bit is set for the variable.
Communication with %s timed out. Timeout happened %d times	Communication to the *%s* driver could not be established after *%d* times within 60 seconds.

10.3 Check list

Problem	Diagnostics	Reason
Values can be read or written by the controller.	The controller can be contacted by 'pinging'?	▶ The controller is not connected to the power supply or the network. ▶ The PC is not connected to the network. ▶ The controller is connected but is in a different subnetwork and the network gateway is not entered in the controller or the subnetmask is not set correctly. ▶ Is the firewall activated? Port 801(TC2x)/851(TC3x) is used for communication by default; add it to the exceptions. Enter accordingly for individual port numbers.

Problem	Diagnostics	Reason
	The controller can be contacted by 'pinging'?	▸ The communication parameter AMS-NET ID is not set correctly? ▸ The port must be set according to the configuration of the controller. ▸ The net address in the addressing of the variable does not correspond to the connection ID in the driver. ▸ The driver configuration file was not transferred to the target computer? ▸ The AMS routes are not configured. Note: The partner station must be entered as a route on he PLC system as well as on the zenon Runtime system.
Certain values cannot be read or written by the controller.	Has an analysis with the Diagnosis Viewer been carried out to see which errors have occurred? See Analysis tool (on page 56) chapter.	▸ Are the variables correctly addressed? IGroup and ▸ Are the correct object types used in the variable? The object types determine the function code to be used in the Modbus telegram.
Incorrect values are displayed.	Has an analysis with the Diagnosis Viewer been carried out to see which errors have occurred? See Analysis tool (on page 56) chapter.	▸ Are the variables correctly addresses and the symbol names correct? ▸ Are the correct data types used? ▸ Is the value calculation correct?

- Is the AMS router correctly configured?
- Are the according remote computers entered vice versa?
- Is the correct port selected?