



COPADATA
do it your way

zenon driver manual

BeckhNG

v.7.20





©2015 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

Distribution and/or reproduction of this document or parts thereof in any form are permitted solely with the written permission of the company COPA-DATA. Technical data is only used for product description and are not guaranteed qualities in the legal sense. Subject to change, technical or otherwise.

Contents

1. Welcome to COPA-DATA help	5
2. BeckhNG	5
3. BECKHNG - Data sheet	6
4. Driver history	7
5. Requirements.....	8
5.1 PC	8
5.2 Control	9
6. Configuration	9
6.1 Creating a driver.....	9
6.2 Settings in the driver dialog	11
6.2.1 General	11
6.2.2 Beckhoff settings	14
7. Creating variables.....	17
7.1 Creating variables in the Editor.....	17
7.2 Addressing.....	20
7.3 Driver objects and datatypes	22
7.3.1 Driver objects	22
7.3.2 Mapping of the data types	24
7.4 Creating variables by importing	26
7.4.1 XML import.....	26
7.4.2 DBF Import/Export	27
7.4.3 Online import	32
7.5 Driver variables	35
8. Driver-specific functions	41
9. Driver commands	44
10. Error analysis.....	46

10.1	Analysis tool	46
10.2	Check list	47

1. Welcome to COPA-DATA help

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 (<mailto:documentation@copadata.com>).

PROJECT SUPPORT

You can receive support for any real project you may have from our Support Team, who you can contact via email at support@copadata.com (<mailto: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 (<mailto: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 and higher-also for TwinCat v3
PLC manufacturer	Beckhoff;

Driver supports:	
Protocol	TC-ADS;
Addressing: Address-based	x
Addressing: Name-based	x
Spontaneous communication	-
Polling communication	x
Online browsing	x
Offline browsing	x
Real-time capable	-
Blockwrite	-
Modem capable	-
Serial logging	-
RDA numerical	x
RDA String	-

Requirements:	
Hardware PC	-
Software PC	The TC-ADS software has to be installed, it is on the installation CD. TwinCat may not be installed on the same computer. The use of TwinCAT CP software is strongly recommended.
Hardware PLC	TwinCAT Soft PLC on PC and CE
Software PLC	-
Requires v-dll	x

Platforms:	
Operating systems	Windows CE 6.0, Embedded Compact 7; Windows 7, 8, 8.1 Server 2008R2, Server 2012, Server 2012R2;
CE platforms	x86; ARM;

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 41) 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

The Beckhoff TwinCAT ADS Communication Library has to be installed on the PC. (TcAdsDll.dll) This software is distributed on the zenon CD. Copy the driver file BeckhNG.exe and the file BeckhNGV.dll to the current zenon directory, unless they are already there.

The single PLC stations have to be notified to the operating system. The stations can be defined with the TwinCAT AMS router. For each station the IP address, the AMS NET-ID, the port number and the communication channel have to be entered. On computers where TwinCAT is not used, the AMS router has to be installed with a minimal installation of TwinCAT (TwinCAT CP). As an alternative, the operating system can be notified about the stations with the tool "TcAmsRemote Mgr.exe". In this case however, the net timeout of the network will be used instead of the configurable AMS timeout. 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 (main.chm::/15247.htm) of the online manual.

6.1 Creating a driver

In order to create a new driver:

1. Right-click on **Driver** in the Project Manage and select **Driver new** in the context menu.

2. In the following dialog the control system offers a list of all available drivers.



3. Select the desired driver and give it a name:
 - The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.
 - 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 with **OK**. In the following dialog the single configurations of the drivers are defined.

Only the respective required drivers need to be loaded for a project. Later loading of an additional driver is possible without problems.



Information

For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:

- ▶ Internal
- ▶ MathDr32
- ▶ SysDrv.

▶

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.



Parameters	Description
Mode	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> ▶ Hardware: <p>A connection to the control is established.</p> ▶ Simulation static <p>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.</p> ▶ Simulation - counting <p>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.</p> ▶ Simulation - programmed <p>N communication is established to the PLC. 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 a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).</p>
Keep update list in the memory	<p>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.</p>
Output can be written	<p>Active: Outputs can be written.</p> <p>Inactive: Writing of outputs is prevented.</p> <p>Note: Not available for every driver.</p>
Variable image remanent	<p>This option saves and restores the current value, time stamp and the states of a data point.</p> <p>Fundamental requirement: The variable must have a valid value and time stamp.</p>

	<p>The variable image is saved in mode hardware if:</p> <ul style="list-style-type: none"> ▶ one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active <p>The variable image is always saved if:</p> <ul style="list-style-type: none"> ▶ the variable is of the object type Driver variable ▶ the driver runs in simulation mode. (not programmed simulation) <p>The following states are not restored at the start of the Runtime:</p> <ul style="list-style-type: none"> ▶ SELECT(8) ▶ WR-ACK(40) ▶ WR-SUC(41) <p>The mode Simulation - programmed at the driver start is not a criterion in order to restore the remanent variable image.</p>
Stop on Standby Server	<p>Setting for redundancy at drivers which allow only on communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade.</p> <p>Attention: If this option is active, the gapless archiving is no longer guaranteed.</p> <p>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 (statusverarbeitung.chm::/24150.htm) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.</p> <p>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.</p>
Global Update time	<p>Active: The set Global update time in ms is used for all variables in the project. The priority set at the variables is not used.</p> <p>Inactive: The set priorities are used for the individual variables.</p>
Priority	<p>The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.</p> <p>The allocation to the variables takes place separately in the settings of the variable properties.</p> <p>The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities.</p>

Thus the communication load is distributed better.

Attention: Priority classes are not supported by each driver. For example, drivers that communicate spontaneously do not support it.

CLOSE DIALOG

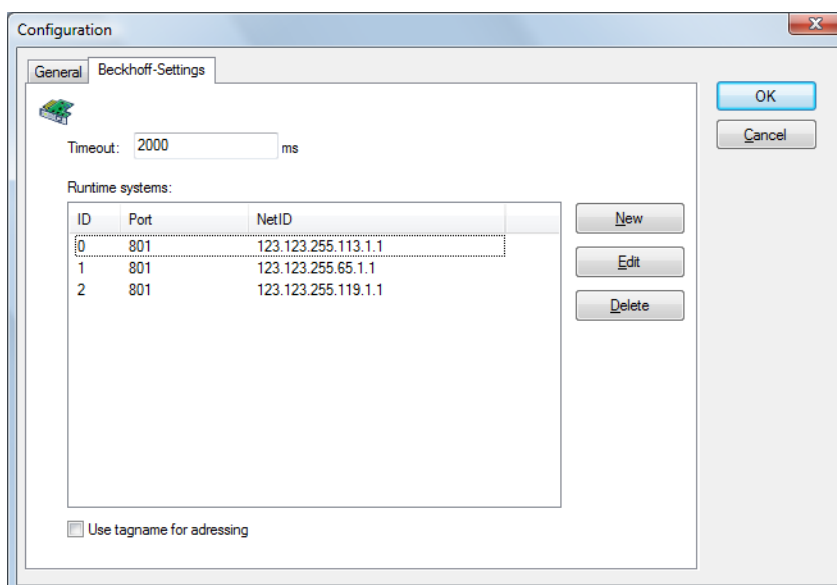
Parameters	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

UPDATE TIME FOR CYCLICAL DRIVERS

The following applies for cyclical drivers:

For **Set value**, **Advising** of variables and **Requests**, a read cycle is immediately triggered for all drivers - regardless of the set update time. This ensures that the value is immediately available for visualization after writing. Update times can therefore be shorter than pre-set for cyclical drivers.

6.2.2 Beckhoff settings



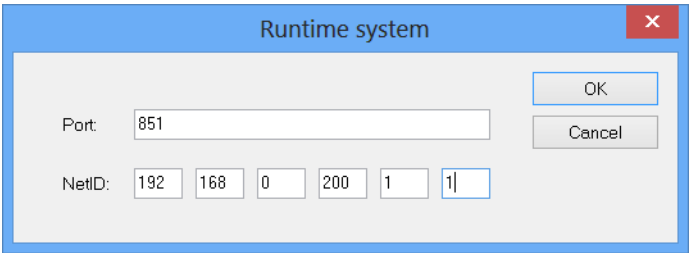
Parameters	Description
Timeout	<p>Entry of the AMS timeout time in milliseconds. The timeout can be set to 2000 ms for AMS routers.</p> <p>If there is an error during the communication with the TwinCAT PLC (e.g. the PLC does not respond), the communication will be interrupted after that time and the driver status will be set to invalid.</p> <p>Refer to chapter Requirements (on page 8).</p>
Runtime systems	<p>This list displays all defined runtime systems (ID, Port and NetID).</p> <p>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.</p> <p>The settings can be changed with the buttons New, Edit and Delete.</p>
Identification contains symbol name	<p>Defines from where the variable addressing via symbol name receives its symbol information:</p> <ul style="list-style-type: none"> ▶ Active: Symbol information come from the identification of the zenon variables. For 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 . ▶ Inactive: Symbol information come from the variable names. <p>Note: This setting is also considered for the online import. For details see Addressing (on page 20) chapter.</p>
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.

CREATE/EDIT RUNTIME SYSTEM

To create a runtime system:

1. Click on the **New** button in the Beckhoff settings tab.
2. The dialog for configuration is opened.

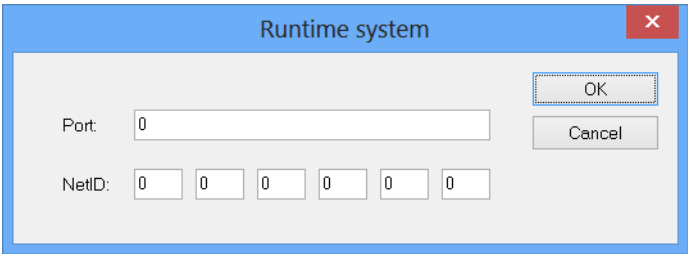
3. Enter **Port** and **NetID**.



To edit an entry:

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

RUNTIME SYSTEM DIALOG



Parameters	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 controllers (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

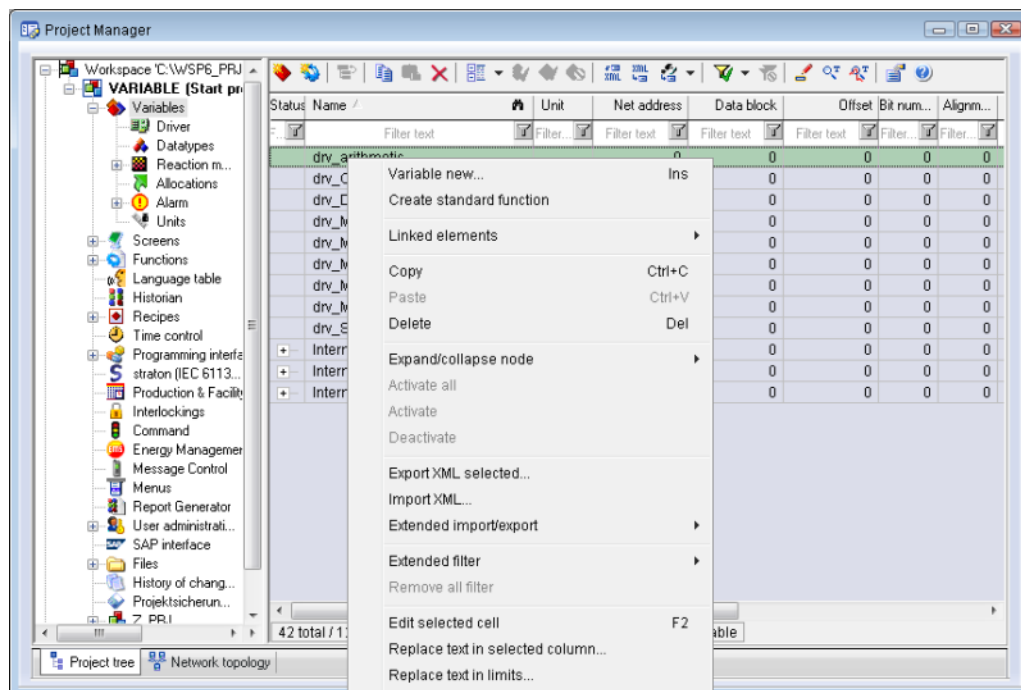
Variables can be created:

- ▶ as simple variables
- ▶ in arrays (main.chm::/15262.htm)
- ▶ as structure variables (main.chm::/15278.htm)

VARIABLE DIALOG

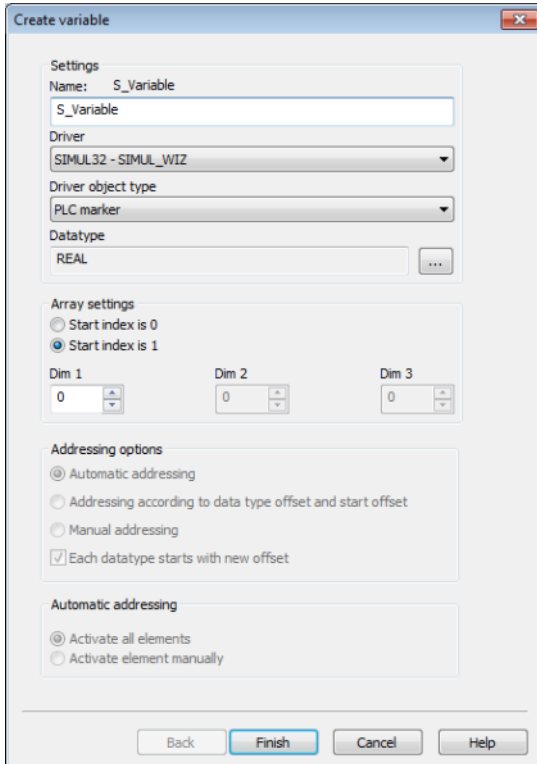
To create a new variable, regardless of which type:

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



2. The dialog for configuring variables is opened
3. configure the variable

4. The settings that are possible depends on the type of variables



The screenshot shows the 'Create variable' dialog box with the following settings:

- Settings**
 - Name: 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

Buttons at the bottom: Back, Finish, Cancel, Help.

Property	Description
Name	<p>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.</p> <p>Maximum length: 128 Zeichen</p> <p>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.</p> <p>Note: For some drivers, the addressing is possible over the property Symbolic address, as well.</p>
Driver	<p>Select the desired driver from the drop-down list.</p> <p>Note: If no driver has been opened in the project, the driver for internal variables (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) is automatically loaded.</p>
Driver object type (cti.chm::/28685.htm)	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 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.

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)
Identification	Symbol name with station prefix, for example "S2_SymName" (see also the information on addressing using a name in relation to this)
Net address	ID of the according station from the driver configuration. If no station with the according AmsNetID exists, a new station is automatically created. (also see "Beckhoff settings")
Data type	The datatype is entered according to IEC.
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.
BitSize	The BitSize from the PLC
Identification	The comment is used as the identification. All line breaks and tabs are removed.

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 14), 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	R / W	WORD, LREAL, BOOL, DWORD, DINT, DATE_AND_TIME, REAL, UDINT, INT, DATE, TOD, UINT, SINT, USINT, TIME, STRING	
Driver variable	35	R / W	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the statistical analysis of communication. Find out more in the chapter about the Driver variables (on page 35)

EXAMPLES FOR ALL POSSIBLE IEC DATA TYPES

On the PLC side	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
INT16	INT
LREAL	LREAL
REAL	REAL
SINT	SINT
STRING	STRING
TIME	TIME
TOD	TOD
TIME_OF_DAY	TOD
UDINT	UDINT
UINT	UINT

UINT16	UINT
USINT	USINT
WORD	WORD

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 FOR 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 LREAL REAL SINT STRING TIME TOD TIME_OF_DAY UDINT UINT	Y	Y	

		UINT16			
		USINT			
		WORD			

Data type: The property **Data type** is the internal numerical name 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 (main.chm::/13028.htm) manual in the Variables (main.chm::/13045.htm) section.

7.4.1 XML import

For the import/export of variables the following is true:

- ▶ The import/export must not be started from the global project.
- ▶ The start takes place via:
 - Context menu of variables or data typ in the project tree
 - or context menu of a variable or a data type
 - or symbol in the symbol bar variables



Attention

When importing/overwriting an existing data type, all variables based on the existing data type are changed.

Example:

There is a data type XYZ derived from the type `INT` with variables based on this data type. The XML file to be imported also contains a data type with the name XYZ but derived from type `STRING`. If this data type is imported, the existing data type is overwritten and the type of all variables based on it is adjusted. I.e. the variables are now no longer `INT` variables, but `STRING` variables.

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 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 export 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 there 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

Description	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the MAX_LAENGE entry in project.ini .
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (field/column must be entered manually). The length can be limited using the MAX_LAENGE entry in project.ini .
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 project.ini .
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the 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	Bus address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADDRESS	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 Recipe Group Manager
LES_SCHR	R	1	Write-Read-Authorization

			0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTYP 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
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.
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 DEFINITION

Limit definition for limit values 1 to 4, and status 1 bis 4:

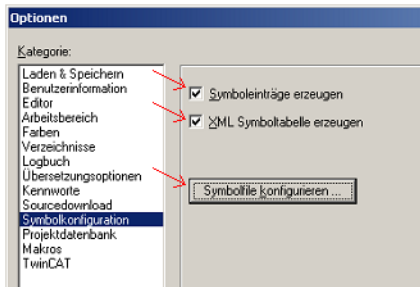
Description	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	technical value or ID number of a linked variable for a dynamic limit (see VARIABLEx) (if VARIABLEx is 1 and here it is -1, the existing variable linkage is not overwritten)
SCHWWERT1	F	16	Threshold value for limit
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)
QUITTTIER1	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 (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 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.

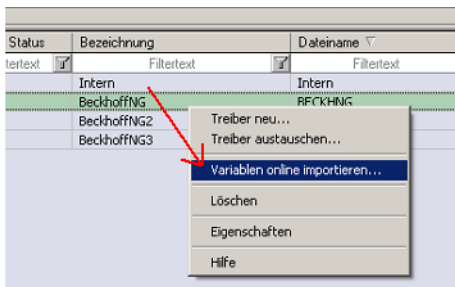
7.4.3 Online import

IMPORT VARIABLES ONLINE

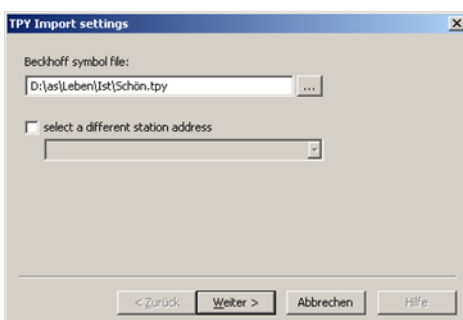
Variables can be imported into zenon via the symbol file (*.tpy). For this, you have to make sure to activate the checkboxes **Create symbol entries** and **Create XML symbol table** under Options/Symbol configuration in the according TwinCAT project. The symbol file also has to be configured accordingly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



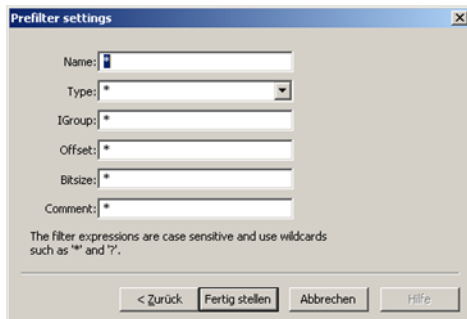
In zenon, the variables can be easily created and changed with the online import. To do this, right-click on the desired driver in the zenon Editor and select the command **Import variables online...** from the context menu.



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.

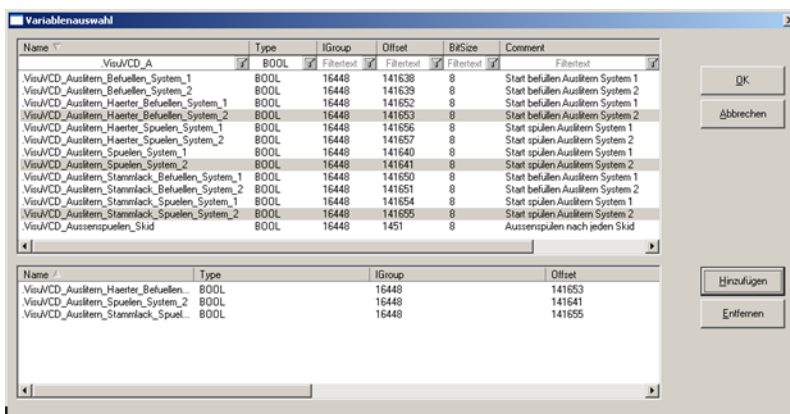


Attention: If there is no AMS net ID in the .tpy file, a station with the AMS net ID "0.0.0.0.0" is created automatically. During import, manually select the correct AMS net ID or amend the AMS Net ID in the driver configuration. A .tpy file from TwinCAT 3.x does not contain an AMS net ID.



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.



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 14):

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 Driver variables

The driver kit implements a number of driver variables. These are divided into:

- ▶ Information
- ▶ Configuration
- ▶ Statistics and
- ▶ Error message

The definitions of the variables defined in the driver kit are available in the import file `drvvar.dbf` (on the CD in the directory: `CD_Drive:/Predefined/Variables`) and can be imported from there.

Note: Variable names must be unique in zenon. If driver variables are to be imported from `drvvar.dbf` again, the variables that were imported beforehand must be renamed.



Information

Not every driver supports all driver variants.

For example:

- ▶ Variables for modem information are only supported by modem-compatible drivers
- ▶ Driver variables for the polling cycle only for pure polling drivers
- ▶ Connection-related information such as ErrorMessage only for drivers that only edit one connection at 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

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 <code>driver stop</code> , the variable has the value <code>TRUE</code> and an OFF bit. After the driver has started, the variable has the value <code>FALSE</code> and no OFF bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If <code>TRUE</code> , 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 <code>SrvDrvVarApplyCom</code> 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 <code>SrvDrvVarApplyModem</code> . 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.
Baud rate	UDINT	10	Baud rate of the serial interface.
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.
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.

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 bus controller (e.g. BC9000) the communication first is tried via a handle. As a bus controller of this type does not support this type of communication, the communication will be delayed. Instead of the Beckhoff NG driver here the Beckhoff BC driver especially developed for this PLC type should be used.

BLOCK ARRAYS

Block arrays can be read.

- ▶ Name-based controllers: The symbol name without indexes is used.
For example: For example **MAIN.MyArray**. Square brackets cannot be used.
- ▶ Address-based controllers: The address of the first index is used.

The number of indexes must correspond to that in the controller. For example:

- ▶ For **MyArray**: **ARRAY [0..99] OF INT;**

- ▶ in the zenon 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 logging.

EXTENDED ERROR FILE

The driver does not write an extended error file.

REDUNDANCY

Possible.

RDA

PROBLEM

The RDA mechanism actually needs a coherent block from zenon in the controller, which can be addressed via an offset. RDA is therefore generally not possible for drivers with symbolic addressing.

SOLUTION

The BeckhNG 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 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 (archivserver.chm::/28257.htm) 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

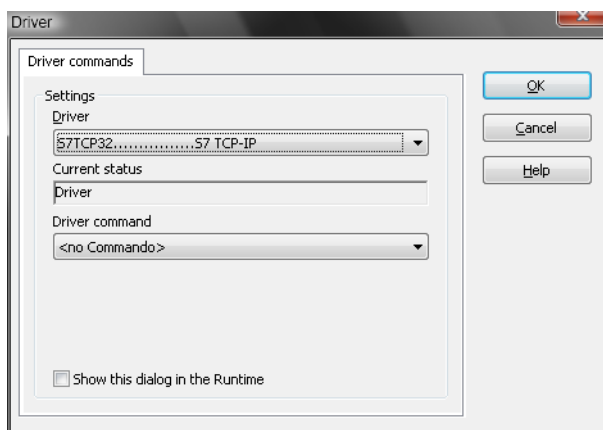
If, for example, an `RDA_4` type `RDA_Test` variable is created in the TwinCAT project, then a variable with the symbolic address `RDA_Test.Trigger` is created in zenon, and `HD active` and `Updated` values are set. If the variable changes value from 0 to 1, the `.Trigger` part of the symbol name `RDA_Test.Trigger` 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 commands

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.

Driver commands are used to influence drivers using zenon; start and stop for example. The engineering is implemented with the help of function `Driver commands`. To do this:

- ▶ create a new function
- ▶ select *Variables* -> *Driver commands*
- ▶ The dialog for configuration is opened



Parameters	Description
Drivers	Drop-down list with all drivers which are loaded in the project.
Current state	Fixed entry which has no function in the current version.
Driver commands	Drop-down list for the selection of the command.
▶ Start driver (online mode)	Driver is reinitialized 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 <code>switched off (OFF; Bit 20)</code> .
▶ 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	Enter driver-specific commands. Opens input field in order to enter a command.
▶ Activate driver write set value	Write set value to a driver is allowed.
▶ Deactivate driver write set value	Write set value to a driver is prohibited.
▶ Establish connection 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)
Show this dialog in the Runtime	The dialog is shown in Runtime so that changes can be made.

DRIVER COMMANDS IN THE NETWORK

If the computer, on which the **driver command** function is executed, is part of the zenon network, additional actions are carried out. A special network command is sent from the computer to the project server, which 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 (main.chm::/12464.htm) program that was also installed with zenon. You can find it under *Start/All programs/zenon/Tools 7.20 -> Diagviewer*.

zenon driver log all errors in the log files. The default folder for the log files is subfolder `LOG` in directory `ProgramData`, example:

`C:\ProgramData\COPA-DATA\LOG`. Log files are text files with a special structure.

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 currently created entries live
- ▶ customize the logging settings
- ▶ change the folder in which the log files are saved

Note:

1. In Windows CE even errors are not logged per default due to performance reasons.
2. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.
3. 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.
4. If you only use **Error logging**, the problem description is in column **Error text**. For other diagnosis level the description is in column **General text**.
5. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** and/or **Error code** and/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.
6. 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**.

You can find further information on the Diagnosis Viewer in the Diagnose Viewer (main.chm::/12464.htm) chapter.

10.2 Check list

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