



zenon
by COPA-DATA

zenon driver manual

ALLANBNT

v.8.20



© 2020 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 properties in the legal sense. Subject to change, technical or otherwise.

Contents

1	Welcome to COPA-DATA help	5
2	ALLANBNT	5
3	ALLANBNT - data sheet.....	5
4	Driver history.....	7
5	Requirements	8
5.1	PC.....	8
5.2	Control	8
6	Configuration	9
6.1	Creating a driver.....	9
6.2	Create driver RSLinx.....	13
6.3	Settings in the driver dialog	16
6.3.1	General	16
6.3.2	Settings	20
6.3.3	Unsolicited Messages configuration	21
6.3.4	ControlLogix Gateway Routing.....	24
7	Creating variables	28
7.1	Creating variables in the Editor	28
7.2	Addressing.....	32
7.3	Driver objects and datatypes	34
7.3.1	Driver objects.....	34
7.3.2	Mapping of the data types.....	36
7.4	Creating variables by importing.....	37
7.4.1	XML import.....	37
7.4.2	DBF Import/Export.....	38
7.5	Communication details (Driver variables).....	43
8	Driver-specific functions	49
9	Driver command function.....	50
10	Error analysis	55
10.1	Analysis tool.....	55

10.2 Driver monitoring	56
10.3 Check list	57

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 ALLANBNT

This driver connects to one or more PLCs in a direct connection (COM interface or network).

3 ALLANBNT - data sheet

General:	
Driver file name	ALLANBNT.exe

General:	
Driver name	Allen Bradley RS-Linx driver
PLC types	Allen Bradley PLC types PLC2, PLC3, PLC5, PLC5250, SLC500;
PLC manufacturer	Allen-Bradley; Rockwell

Driver supports:	
Protocol	DataHighway Plus; DeviceNet; DataHighway; DF1; TCP/IP; ControlNet
Addressing: Address-based	Address based
Addressing: Name-based	--
Spontaneous communication	--
Polling communication	X
Online browsing	--
Offline browsing	--
Real-time capable	--
Blockwrite	--
Modem capable	--
RDA numerical	--
RDA String	--
Hysteresis	--
extended API	X
Supports status bit WR-SUC	--
alternative IP address	--

Requirements:	
Hardware PC	Network or serial connections

Requirements:	
Software PC	Rockwell RSLinx (ab v1.6) required. Licensed version of RS Linx required. (at least RS Linx Classic Gateway) RS Linx Classic Lite does not work.
Hardware PLC	--
Software PLC	--
Requires v-dll	--

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
8/14/2008	500	Created driver documentation

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

HARDWARE

There must be connectors for network or serial connections as well as Serial, BUS and RJ45 cables.

SOFTWARE

Copy the driver file (*.exe) into the current program directory (if they do not already exist there) and enter into TREIBER_EN.XML with the tool driverinfo.exe.

Additionally the software RSLinx from Rockwell Automation is needed.

5.2 Control

HARDWARE

PLC2, PLC3, PLC5, PLC5250, SLC500 and network or serial connectors required.

SOFTWARE

PLC can be configured with tools (Allan Bradley).

6 Configuration

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

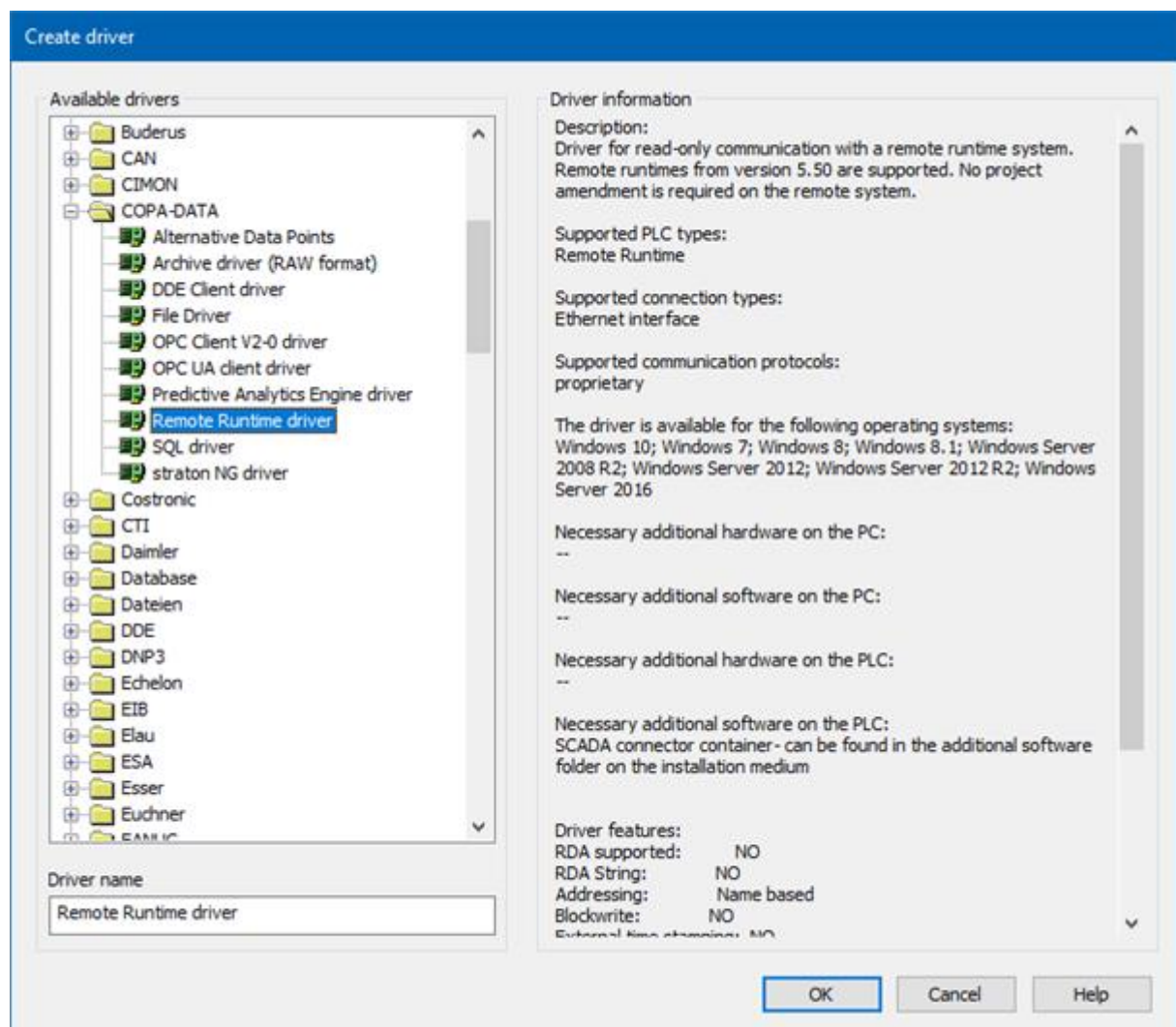


Information

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.



Parameter	Description
Available drivers	<p>List of all available drivers.</p> <p>The display is in a tree structure: [+] expands the folder structure and shows the drivers contained therein. [-] reduces the folder structure</p> <p>Default: <i>No selection</i></p>
Driver name	<p>Unique Identification of the driver.</p> <p>Default: <i>empty</i></p> <p>The input field is pre-filled with the pre-defined Identification after selecting a driver from the list of available drivers.</p>
Driver information	<p>Further information on the selected driver.</p> <p>Default: <i>empty</i></p> <p>The information on the selected driver is shown in this area after selecting a driver.</p>

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

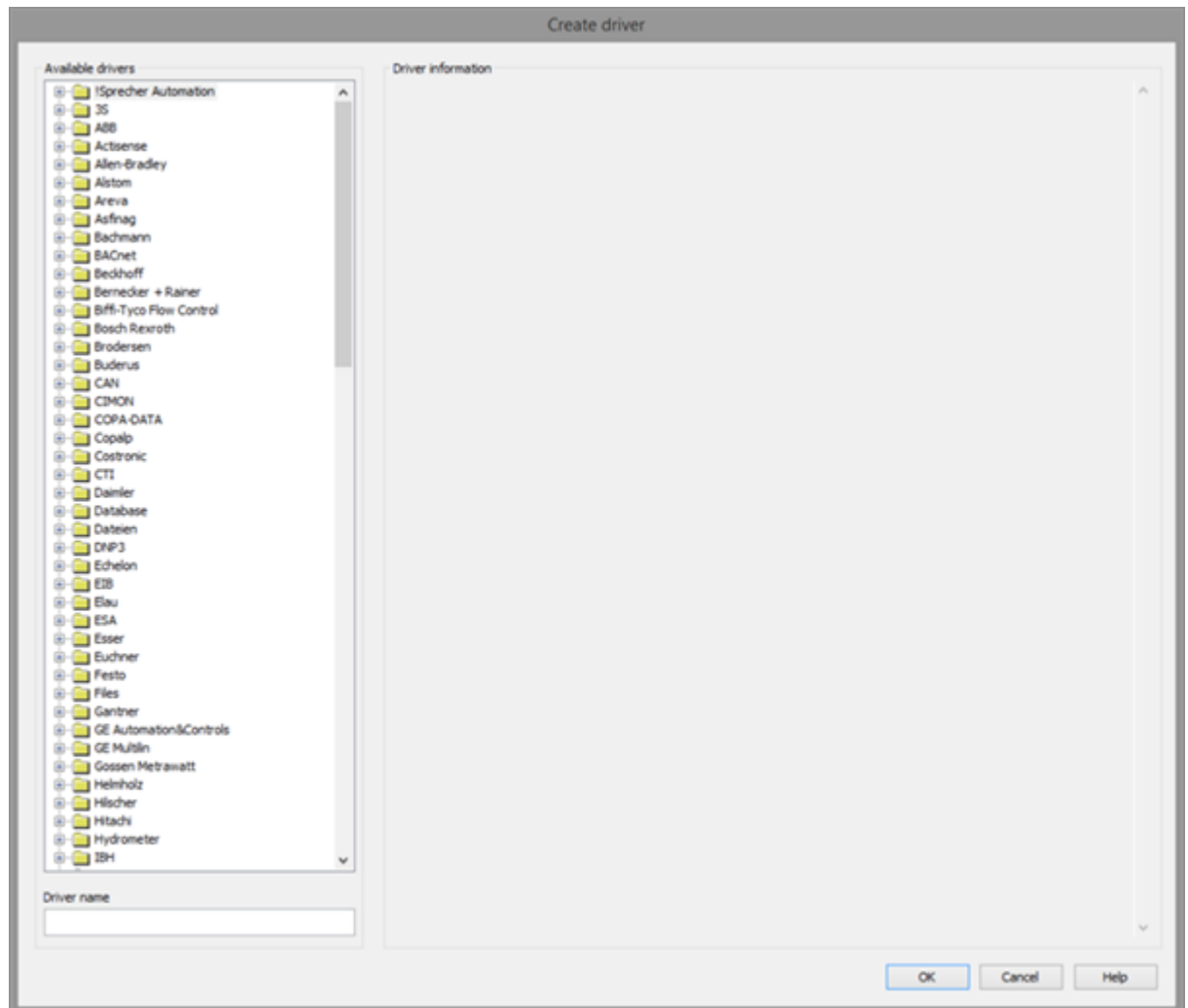
In order to create a new driver:

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**:

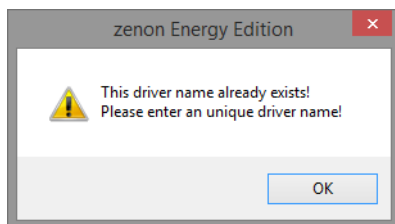
- ▶ The **Driver name** must be unique. If a driver is used more than once in a project, a new name has to be given each time. This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
- ▶ The **Driver name** is part of the 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 PROJECT

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

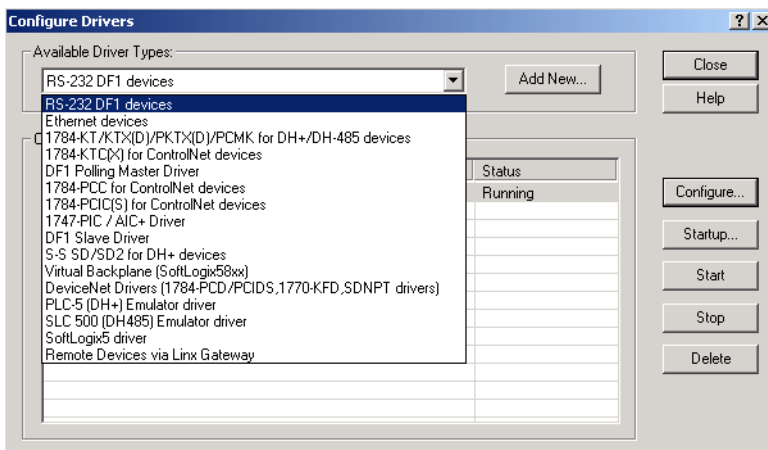
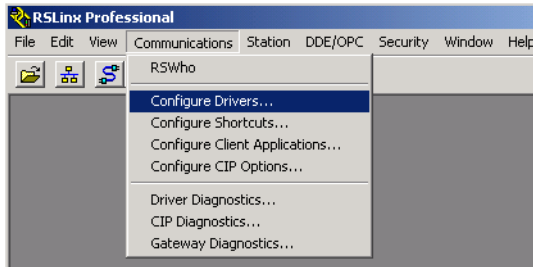
- ▶ **Intern**
- ▶ **MathDr32**
- ▶ **SysDrv**

Information

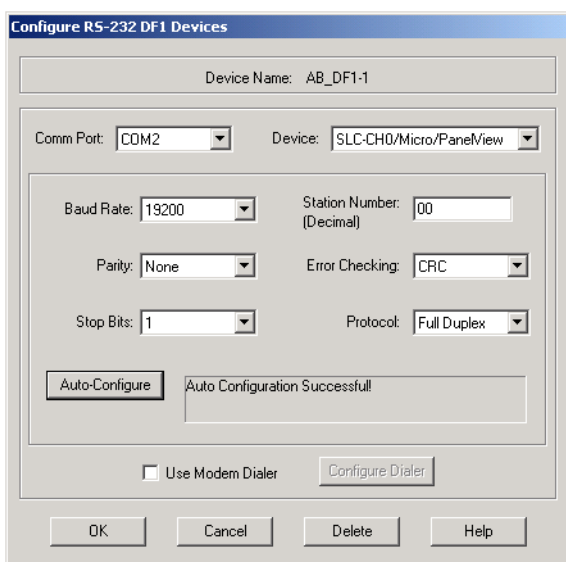
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

6.2 Create driver RSLinx

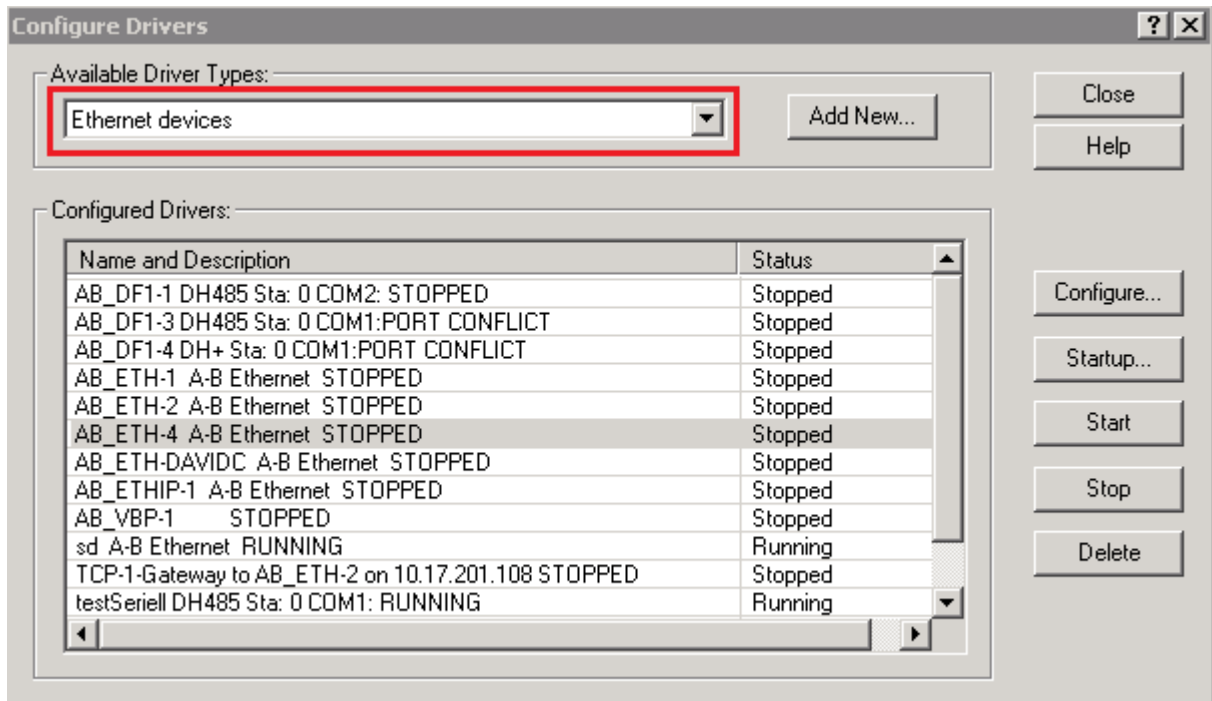
SERIAL CONFIGURATION



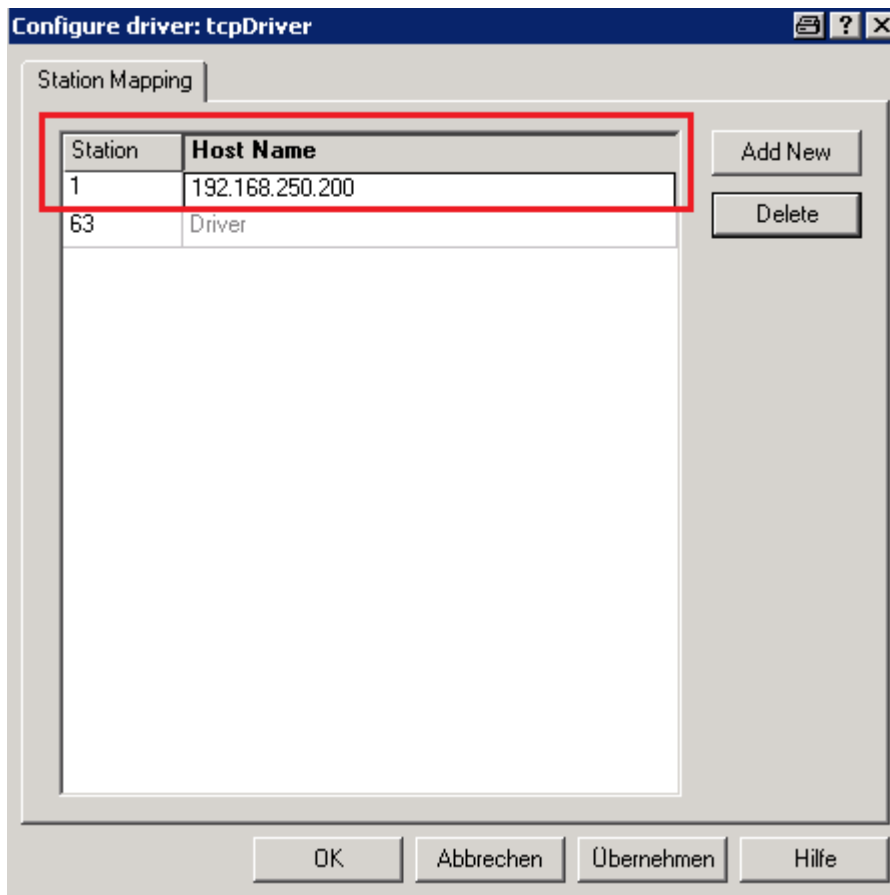
Select the according driver from the menu **Available Driver Types**. Add it with **Add New..**; this will also create a name for the driver. To open the driver configuration, select the driver in the list and press the button **Configure**.



TCP/IP CONFIGURATION



Select the **Ethernet devices** entry from the **Available Driver Types:** menu. Click on **Add New...** and give the driver a name. Then highlight the driver you created and click on **Configure** to configure it.

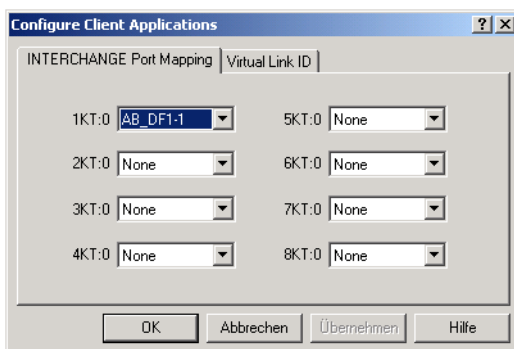
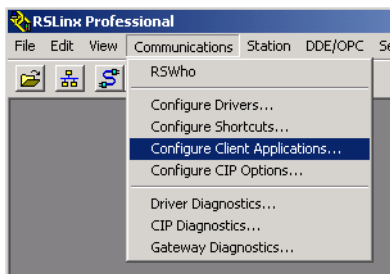


For **Host name** , enter the IP address of the controller to which the driver should connect, and for **Station** , enter a desired numeric station address.

Attention

If the PLC is connected, RSLinx will be able to read the correct parameters from it.

Otherwise, you will have to enter the connection parameters here.

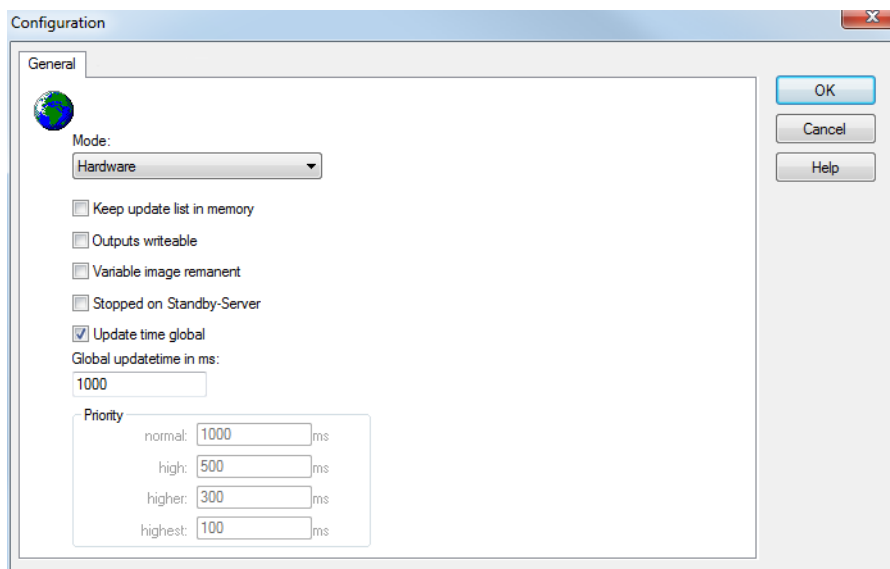


6.3 Settings in the driver dialog

You can change the following settings of the driver:

6.3.1 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.



Option	Description
Mode	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> ▶ <i>Hardware:</i> A connection to the control is established. ▶ <i>Simulation - static:</i> 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. ▶ <i>Simulation - counting:</i> 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. ▶ <i>Simulation - programmed:</i> No 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.
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	<ul style="list-style-type: none"> ▶ <i>Active:</i> Outputs can be written. ▶ <i>Inactive:</i> Writing of outputs is prevented.

Option	Description
Variable image remanent	<p>Note: Not available for every driver.</p> <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 hardware mode if one of these statuses is active:</p> <ul style="list-style-type: none"> ▶ User status <i>M1 (0)</i> to <i>M8 (7)</i> ▶ <i>REVISION(9)</i> ▶ <i>AUS(20)</i> ▶ <i>ERSATZWERT(27)</i> <p>The variable image is always saved if:</p> <ul style="list-style-type: none"> ▶ the variable is of the Communication details object type ▶ 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"> ▶ <i>SELECT(8)</i> ▶ <i>WR-ACK(40)</i> ▶ <i>WR-SUC(41)</i> <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 one 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> <ul style="list-style-type: none"> ▶ <i>Active:</i> 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

Option	Description
	<p>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.</p> <p>Default: <i>inactive</i></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>Setting for the global update times in milliseconds:</p> <ul style="list-style-type: none"> ▶ <i>Active:</i> The set Global update time is used for all variables in the project. The priority set at the variables is not used. ▶ <i>Inactive:</i> The set priorities are used for the individual variables. <p>Exceptions: Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the Spontaneous driver update time section.</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 variables are allocated separately in the settings of the variable properties.</p> <p>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.</p> <p>Attention: Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.</p>

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.

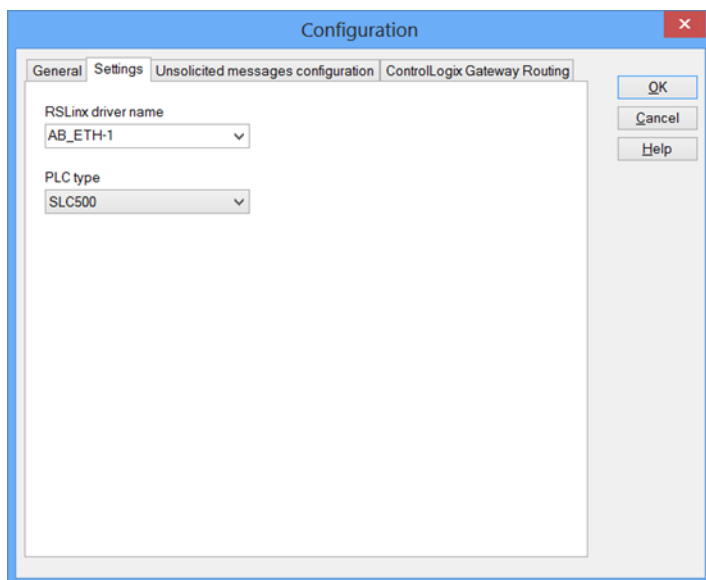
UPDATE TIME FOR SPONTANEOUS DRIVERS

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**.

6.3.2 Settings

Configuration of RSLink driver name and PLC type.



Parameter	Description
RSLinx driver name	Label of the RS-LINX driver. Enter it in the field manually or select it via the drop-down list. It contains the drivers available in RS-LINX.
PLC Type	Type of the used PLC. Select from drop-down list.

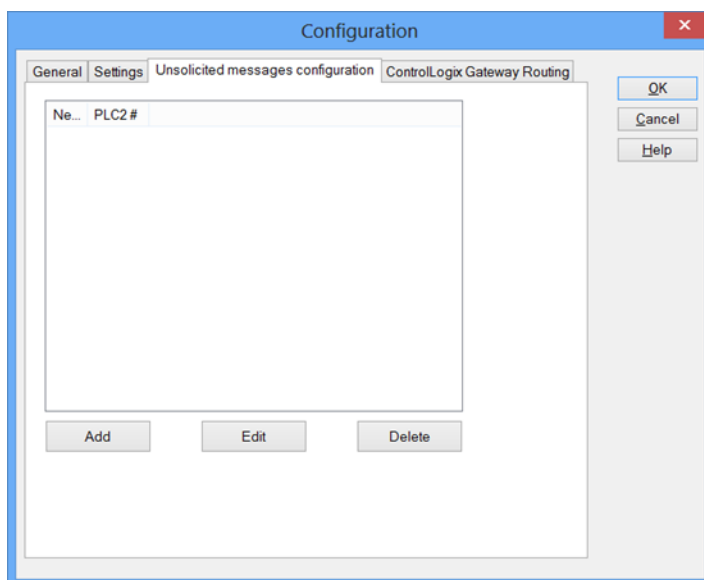
Parameter	Description
	Note: Is ignored when routing (on page 24) is used.
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

A PLC-2 unprotected write command contains only one setting for the addressing - the **data table address**. The mapping of this address to the net address of the variable is carried out in the **Unsolicited messages configuration** (on page 21).

6.3.3 Unsolicited Messages configuration

Messages of type *unsolicited* can be read and written by the zenon RS Linx driver.

To depict received data on the engineered variables, you must map the **data table address** of the PLC2 to the variable address. For each message one entry must be configured.



Parameter	Description
List field	Contains all configured allocations from variable net addresses to PLC2 addresses.
Add	Opens dialog in order to add a new allocation.
Edit	Opens dialog in order to edit the allocation highlighted in the list field.
Delete	Deletes the allocation highlighted in the list field from the list without confirmation request. If several allocations are highlighted, only the topmost

Parameter	Description
	allocation is deleted.
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.

ALLOCATION

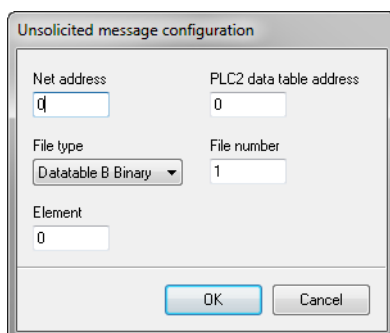
In the PLC program - triggered by the program - ein Unsolicited Message is sent with a **PLC2 unprotected write** command. Target address is the computer with zenon and RS-LINX.

For the allocation of PLC2 addresses to variables the following is true:

- ▶ The same PLC2 address can be allocated to different variables with different net addresses.
- ▶ A pair composed of a net address and a PLC2 address can only be allocated to one variable address.
- ▶ If several variables lie one after the other and if the write command contains data for several variables, these data are automatically depicted in the following variable.

To configure the allocation:

1. click on the button **Add** on tab *Unsolicited MessagesConfiguration*
2. the dialog for configuring an allocation is opened



The dialog box titled "Unsolicited message configuration" contains the following fields:

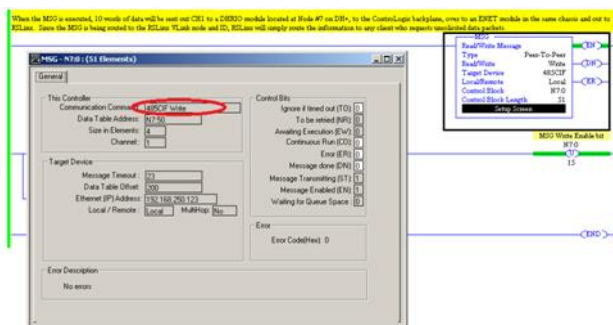
- Net address:** A text input field containing the value "Q".
- PLC2 data table address:** A text input field containing the value "0".
- File type:** A drop-down menu with "Datatable B Binary" selected.
- File number:** A text input field containing the value "1".
- Element:** A text input field containing the value "0".

At the bottom of the dialog are "OK" and "Cancel" buttons.

Parameter	Description
Net address	Network address of variables.
PLC2 data table address	Data table address of the PLC2.
File type	Selection of the type from the drop-down list.
File number	File number.
Element	Element number.

Parameter	Description
OK	Takes over settings, creates entry in list and closes the dialog.
Cancel	Rejects all setting changes and closes the dialog.

In the driver configuration you must create a corresponding unsolicited message for the fitting station number (configured in RS_LINX).



Information

For zenon version < 6.51, the **Update time global** option must be deactivated in the General (on page 16) tab for the Unsolicited Messages to be processed faster than the **Update time global**.

EXAMPLE

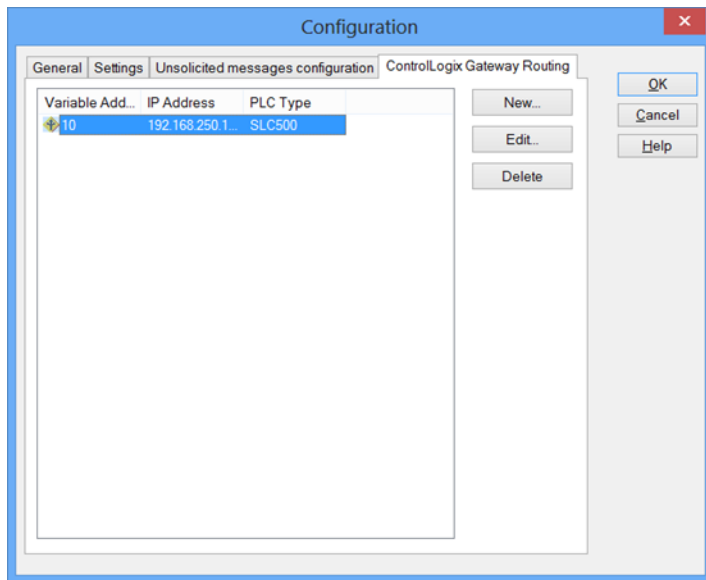
4 integer values - N7:50 to N7:53 - are copied as data in the message. As **PLC2 data table address 200** is used.

As in the RS-LINX configuration the **Data table address 200** is again depicted on integer values N7:50 and following:

- ▶ the corresponding variables (N7:50 - N7:53) can be polled (according to the defined update time)
- ▶ received spontaneous values which were received additionally via Unsolicited Messages can be depicted on this variable

6.3.4 ControlLogix Gateway Routing

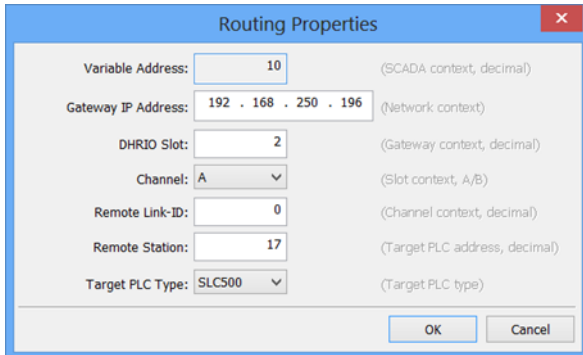
For variables, routing can be configured using DH+. For details, see the **Routing of variables via DH+** (on page 26) chapter.



Parameter	Description
Routings list field	Shows configured routings.
New	Opens the dialog for configuring a new routing.
Edit	Opens the dialog for editing the routing shown in the list.
Delete	Deletes the selected routing from the list.
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.

CREATING OR EDITING ROUTING

Clicking on the **New** or **Edit** button opens the dialog to configure a routing. The required values can be collected with the RSLinx software. You can find details in the RSLinx software documentation.



The **Routing Properties** dialog box contains the following fields and controls:

- Variable Address:** Text box with value 10. (SCADA context, decimal)
- Gateway IP Address:** Text box with value 192 . 168 . 250 . 196. (Network context)
- DHRIO Slot:** Text box with value 2. (Gateway context, decimal)
- Channel:** Dropdown menu with value A. (Slot context, A/B)
- Remote Link-ID:** Text box with value 0. (Channel context, decimal)
- Remote Station:** Text box with value 17. (Target PLC address, decimal)
- Target PLC Type:** Dropdown menu with value SLC500. (Target PLC type)
- Buttons:** OK and Cancel at the bottom right.

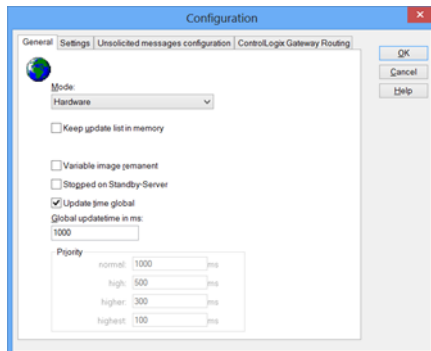
Parameter	Description
Variable Address	Net address of the variable.
Gateway IP Address	IP address of the gateway controller.
DHRIO Slot	Number of the slot in the rack in which the DHRIO module is located.
Channel	Selected channel.
Remote Link ID	ID of the station. <ul style="list-style-type: none"> ▶ 0: Local. ▶ >0: Link ID of the connection configuration in RSLinx
Remote Station	Number of the remote station.
Target PLC Type	Type of target PLC. Note: If a different PLC has been selected in the Settings (on page 20) tab, the PLC selected here is used for routing.
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

6.3.4.1 Configuration of variable routing via DH+

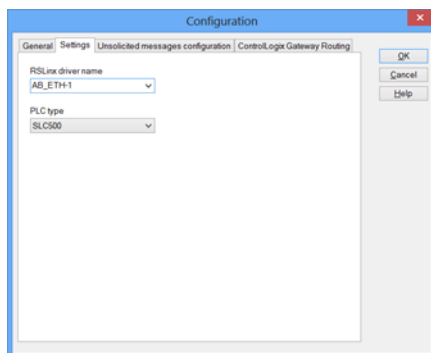
Configuration of the routing for communication with variables that are connected via DH+. You can find information on parameters from RSLinx in the module configuration in the RSLinx software.

To configure the routing:

1. In the driver configuration, select *Hardware* as the **mode** in the **General** tab.

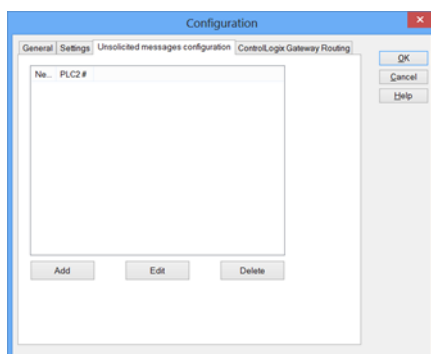


2. Select the **RSLinx driver name** in the Settings tab.

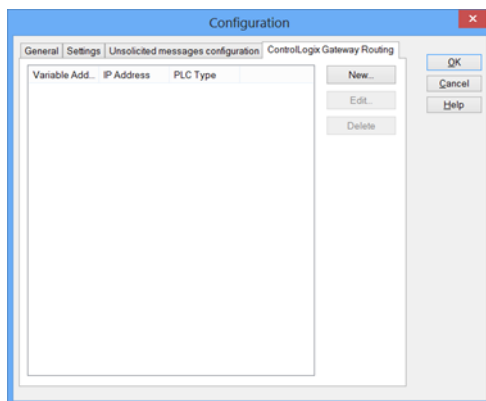


For routing, the PLC type option is taken from the settings in the **ControlLogix Gateway Routing** (on page 24) tab. Settings that are different to this in the **Settings** (on page 20) tab are ignored.

3. Configure the **Unsolicited Messages Configuration** (on page 21) tab.

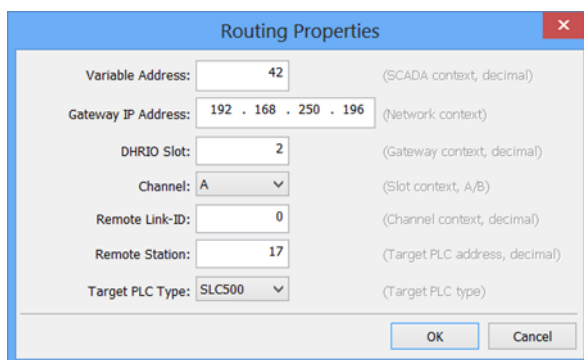


4. Configure the **ControllLogix Gateway Routing** tab.

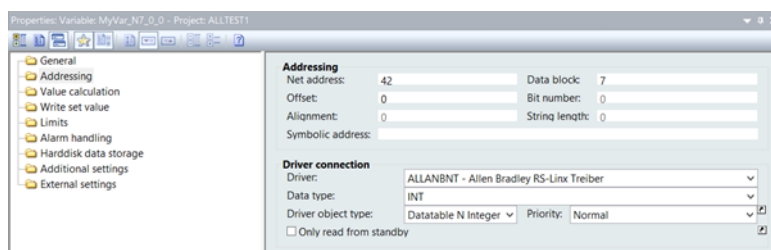


5. Click on the **New** button to configure a new routing.

The dialog for the routing properties is opened. Configure the properties:



- a) Enter, for the **variable address**, the **Net address** of the variables in zenon.



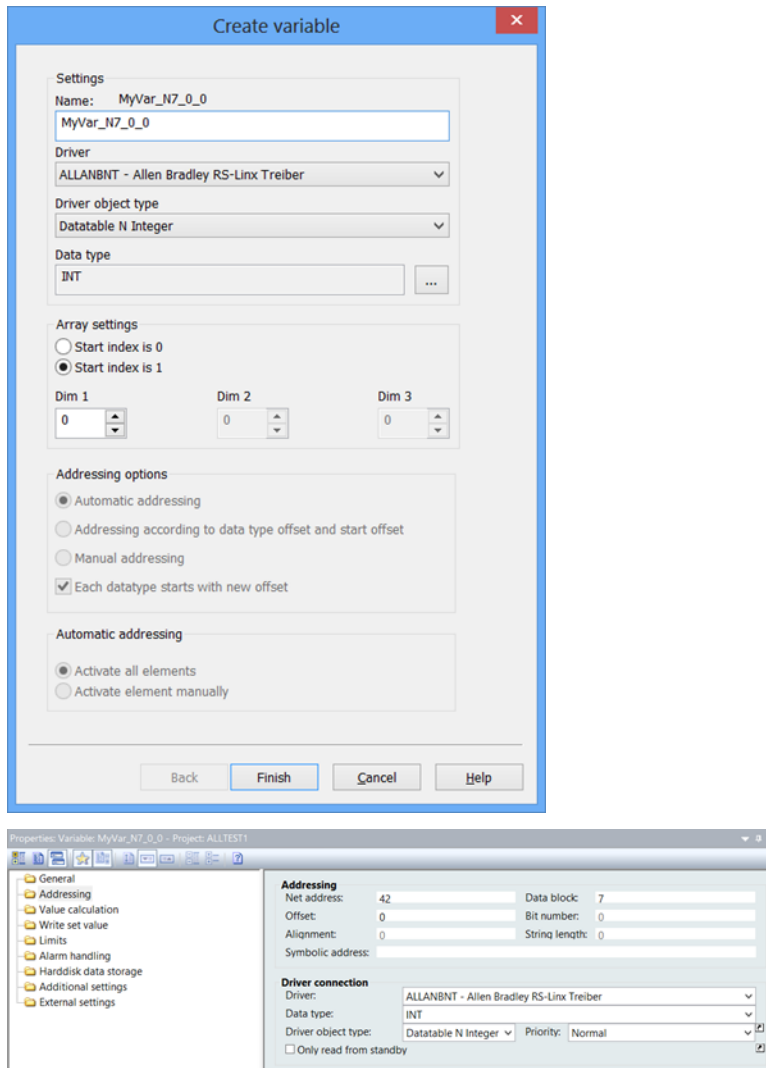
- b) Enter the IP address of the gateway in RSLinx.
- c) Enter the slot in the rack in **DHRIO Slot**.
- d) Enter the **remote link ID**.

If it is the same station as that defined in the gateway IP address option, then set the value to 0.

If routing is via a different station, enter the **link ID** for the corresponding connection.

- e) Enter the address of the target PLC.
- f) Enter the type of the target PLC. This setting is preferred over the setting with the same name in the **Settings** (on page 20) tab.

6. Configure the corresponding variables:



The image shows two screenshots from the Zenon software interface. The top screenshot is the 'Create variable' dialog box. It has a title bar 'Create variable' with a close button. The 'Settings' section includes: 'Name' (MyVar_N7_0_0), 'Driver' (ALLANBNT - Allen Bradley RS-Linx Treiber), 'Driver object type' (Datatable N Integer), and 'Data type' (INT). The 'Array settings' section has radio buttons for 'Start index is 0' and 'Start index is 1' (selected), and three dimension spinners (Dim 1, Dim 2, Dim 3) all set to 0. The 'Addressing options' section has radio buttons for 'Automatic addressing' (selected), 'Addressing according to data type offset and start offset', and 'Manual addressing', plus a checked checkbox 'Each datatype starts with new offset'. The 'Automatic addressing' section has radio buttons for 'Activate all elements' (selected) and 'Activate element manually'. At the bottom are 'Back', 'Finish', 'Cancel', and 'Help' buttons.

The bottom screenshot is the 'Properties' window for the variable 'MyVar_N7_0_0'. It has a title bar 'Properties: Variable: MyVar_N7_0_0 - Project: ALLTEST1'. The left pane shows a tree view with categories: General, Addressing, Value calculation, Write set value, Limits, Alarm handling, Harddisk data storage, Additional settings, and External settings. The right pane shows the 'Addressing' and 'Driver connection' sections. The 'Addressing' section includes: 'Net address' (42), 'Data block' (7), 'Offset' (0), 'Bit number' (0), 'Alignment' (0), 'String length' (0), and 'Symbolic address'. The 'Driver connection' section includes: 'Driver' (ALLANBNT - Allen Bradley RS-Linx Treiber), 'Data type' (INT), 'Driver object type' (Datatable N Integer), 'Priority' (Normal), and a checkbox 'Only read from standby'.

7 Creating variables

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

7.1 Creating variables in the Editor

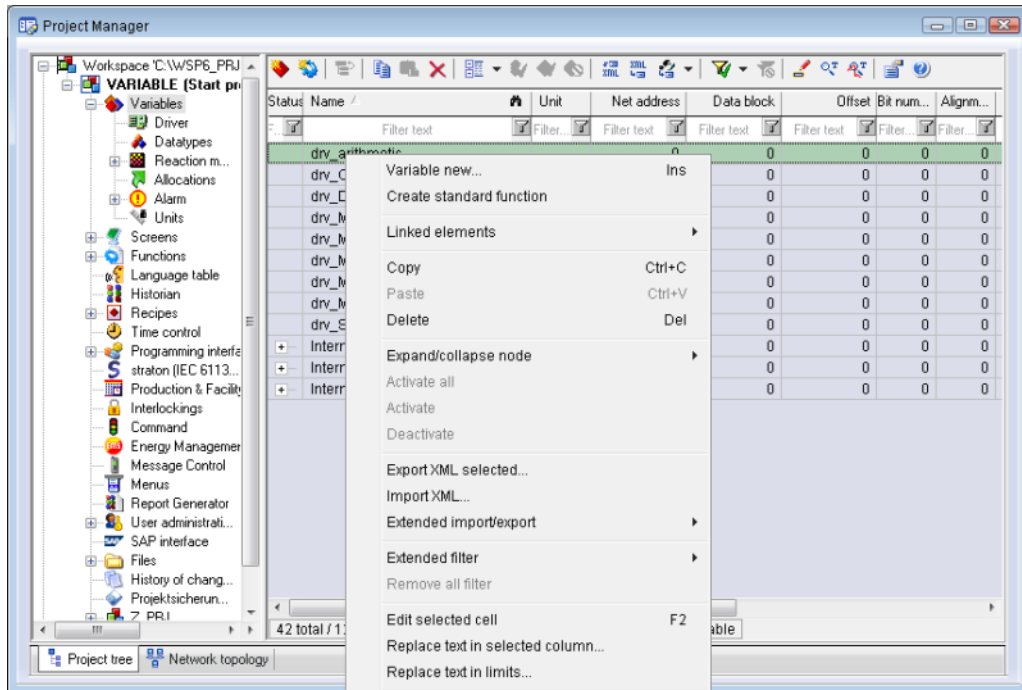
Variables can be created:

- ▶ as simple variables
- ▶ in arrays
- ▶ as structure variables

VARIABLE DIALOG

To create a new variable, regardless of which type:

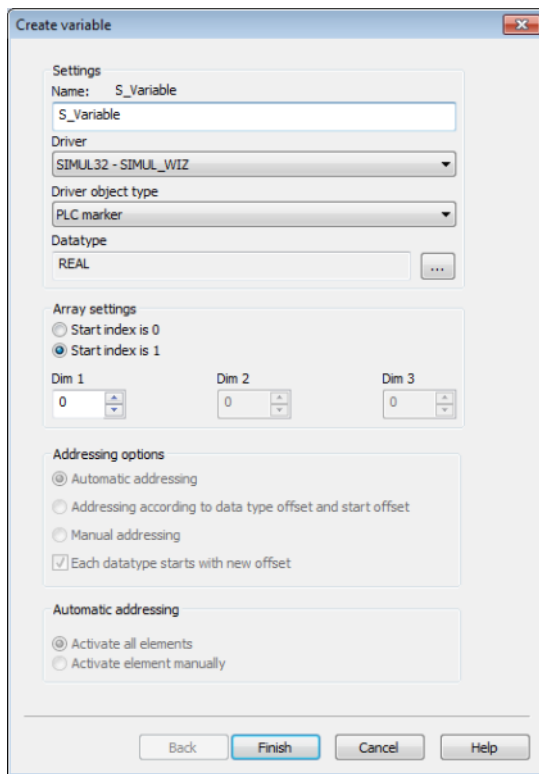
1. Select the **New variable** command in the **Variables** node in the context menu



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



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 characters</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: Some drivers also allow addressing using the Symbolic address property.</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) is automatically loaded.</p>
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
- ▶ KabaDPsServer
- ▶ 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

Property	Description
Name	<p>Freely definable name.</p> <p>Attention: For every zenon project the name must be unambiguous.</p>
Identification	<p>Freely definable identification. E.g. for Resources label, comments, ...</p>
Net address	<p>Bus address or net address of the variable.</p> <p>This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.</p> <p>Corresponds to the Variable Address in RSLink. When using the Ethernet driver in RSLink, addressing is by means of the net address.</p>
Data block	<p>Corresponds to the data tables in the PLC.</p> <p>For variables of object type <i>Extended data block</i>, enter the datablock number here.</p> <p>Adjustable from 0 to 4294967295.</p> <p>You can take the exact maximum area for data blocks from the manual of the PLC.</p>
Offset	<p>Offset of variables. Equal to the memory address of the variable in the PLC. Adjustable from 0 to 4294967295.</p>
Alignment	not used for this driver
Bit number	<p>Number of the bit within the configured offset.</p> <p>Possible entries: 0 to 65535.</p>
String length	<p>Only available for String variables. Maximum number of characters that the variable can take.</p>
Driver connection/Driver	<p>Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.</p>

Property	Description
Object Type	
Driver connection/Data Type	Data type of the variable. Is selected during the creation of the variable; the type can be changed here. Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.
Driver connection/Priority	Assigns a variable a priority for the update time.

ASSIGNMENT OF ADDRESSES OF ALLEN BRADLEY PLCS TO ZENON

AB control units administer addresses in this form: **Data_type_short+numeral:offset/bit number**.
For example: **B3:100/7**

Assignment of data types and addresses:

Data type	Short form	Digit
<i>BINARY</i>	B	3
<i>TIMER</i> (Only DF1)	T	4
<i>COUNTER</i> (Only DF1)	C	5
<i>CONTROL</i> (Only DF1)	R	6
<i>INTEGER</i>	N	7
<i>FLOATING PT</i>	F	8

To transfer these addresses in zenon correctly:

- ▶ Enter the **digit** (3, for example) in the **Data block** property
- ▶ The zenon **Offset** corresponds to the address within this bandwidth
- ▶ Bit number corresponds to **Bit number**

EXAMPLE

The AB address **B3:100/7** corresponds to the following in zenon:

- ▶ **Data block:** 3
- ▶ **Offset:** 100
- ▶ **Bit number:** 7

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:

OBJECTS FOR PROCESS VARIABLES IN ZENON

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
Datatable B Binary	96	X	X	USINT, BOOL, SINT, UINT, INT	
Datatable F Float	97	X	X	REAL	
Datatable I Input	101	X	X	BOOL, INT, SINT, UINT, USINT	Sets Data block to 1.
Datatable N Integer	34	X	X	REAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING	
Datatable O Output	102	X	X	BOOL, INT, SINT, UINT, USINT	Sets Data block to 2.

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
Datatable ST String	98	X	X	<i>STRING</i>	Maximum 82 characters
<i>Communication details</i>	35	X	X	<i>BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING</i>	<p>Variables for the static analysis of the communication; Values are transferred between driver and Runtime (not to the PLC).</p> <p>Note: The addressing and the behavior is the same for most zenon drivers.</p> <p>You can find detailed information on this in the Communication details (Driver variables) (on page 43) chapter.</p>

Key:

X: supported

--: not supported

Object	Read	Write	Comment
B block <i>(Word, Byte, Bool)</i>	X	X	Binary - Data table
N block <i>(DWORD, FLOAT, Word, String, Byte/Char, Bool)</i>	X	X	Integer - Data table

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.

Control	zenon	Data type
	BOOL	8
	USINT	9
	SINT	10
	UINT	2
	INT	1
	UDINT	4
	DINT	3
	ULINT	27
	LINT	26
	REAL	5
	LREAL	6
	STRING	12
	WSTRING	21
	DATE	18
	TIME	17
	DATE_AND_TIME	20
	TOD (Time of Day)	19

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	Character	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

Identification	Type	Field size	Comment
			<p>the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered manually).</p> <p>The length can be limited using the MAX_LAENGE entry in the project.ini file.</p>
KANAL_D	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
TAGNR	C	128	<p>Identification.</p> <p>The length can be limited using the MAX_LAENGE entry in the project.ini file.</p>
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	Net address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	<p>For bit variables: bit address</p> <p>For byte variables: 0=lower, 8=higher byte</p> <p>For string variables: Length of string (max. 63 characters)</p>
ARRAYSIZE	N	16	<p>Number of variables in the array for index variables</p> <p>ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager</p>
LES_SCHR	L	1	<p>Write-Read-Authorization</p> <p>0: Not allowed to set value.</p> <p>1: Allowed to set value.</p>
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

Identification	Type	Field size	Comment
			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
MENTIEFE	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.

Identification	Type	Field size	Comment
			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 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

Identification	Type	Field size	Comment
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 (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.

7.5 Communication details (Driver variables)

The driver kit implements a number of driver variables. These 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 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
LineStateldle	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

Name from import	Type	Offset	Description
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 <i>driver stop</i> , the variable has the value <i>TRUE</i> and an OFF bit. After the driver has started, the variable has the value <i>FALSE</i> 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: <ul style="list-style-type: none"> ▶ 0: Connection OK ▶ 1: Connection failure ▶ 2: Connection simulated Formatting: <Net address>:<Connection status>;...;; A connection is only known after a variable

Name from import	Type	Offset	Description
			<p>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.</p> <p>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.</p>

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	<i>BOOL</i>	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	<i>BOOL</i>	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	<i>BOOL</i>	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	<i>STRING</i>	38	Telephone number, that should be used
ModemHwAdrSet	<i>DINT</i>	39	Hardware address for the telephone number
GlobalUpdate	<i>UDINT</i>	3	Update time in milliseconds (ms).
BGlobalUpdaten	<i>BOOL</i>	4	TRUE, if update time is global
TreiberSimul	<i>BOOL</i>	5	TRUE, if driver in sin simulation mode
TreiberProzab	<i>BOOL</i>	6	TRUE, if the variables update list should be kept in the memory
ModemActive	<i>BOOL</i>	7	TRUE, if the modem is active for the driver
Device	<i>STRING</i>	8	Name of the serial interface or name of the modem

Name from import	Type	Offset	Description
ComPort	<i>UINT</i>	9	Number of the serial interface.
Baudrate	<i>UDINT</i>	10	Baud rate of the serial interface.
Parity	<i>SINT</i>	11	Parity of the serial interface
ByteSize	<i>USINT</i>	14	Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection.
StopBit	<i>USINT</i>	13	Number of stop bits of the serial interface.
Autoconnect	<i>BOOL</i>	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	<i>STRING</i>	17	Current telephone number
ModemHwAdr	<i>DINT</i>	21	Hardware address of current telephone number
RxIdleTime	<i>UINT</i>	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)
WriteTimeout	<i>UDINT</i>	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	<i>UDINT</i>	20	Number of ringing tones before a call is accepted
ReCallIdleTime	<i>UINT</i>	53	Waiting time between calls in seconds (s).
ConnectTimeout	<i>UINT</i>	54	Time in seconds (s) to establish a connection.

STATISTICS

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

Name from import	Type	Offset	Description
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.

Name from import	Type	Offset	Description
RdErrBlockCount	<i>UINT</i>	44	Number of blocks to read when the last reading error occurred.
RdErrHwAdresse	<i>DINT</i>	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	<i>UDINT</i>	46	Block number when the last reading error occurred.
RdErrMarkerNo	<i>UDINT</i>	47	Marker number when the last reading error occurred.
RdErrSize	<i>UDINT</i>	48	Block size when the last reading error occurred.
DrvError	<i>USINT</i>	25	Error message as number
DrvErrorMsg	<i>STRING</i>	30	Error message as text
ErrorFile	<i>STRING</i>	15	Name of error log file

8 Driver-specific functions

The driver supports the following functions:

INI ENTRIES

ZENON6.INI

None

PROJECT.INI

None

LIMITATIONS

Only data tables (datablocks) also existing in the PLC can be accessed.

Supported data tables see driver objects.

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

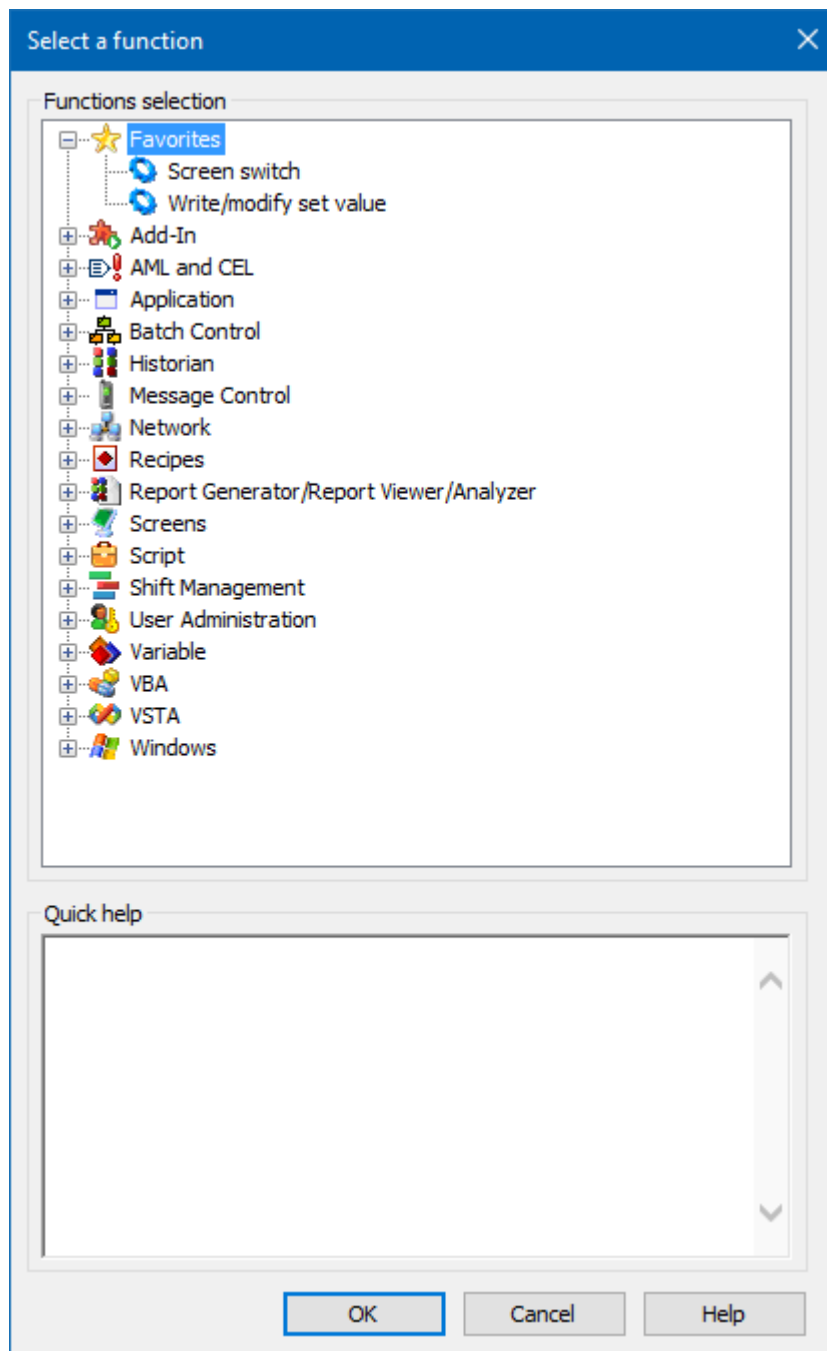
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

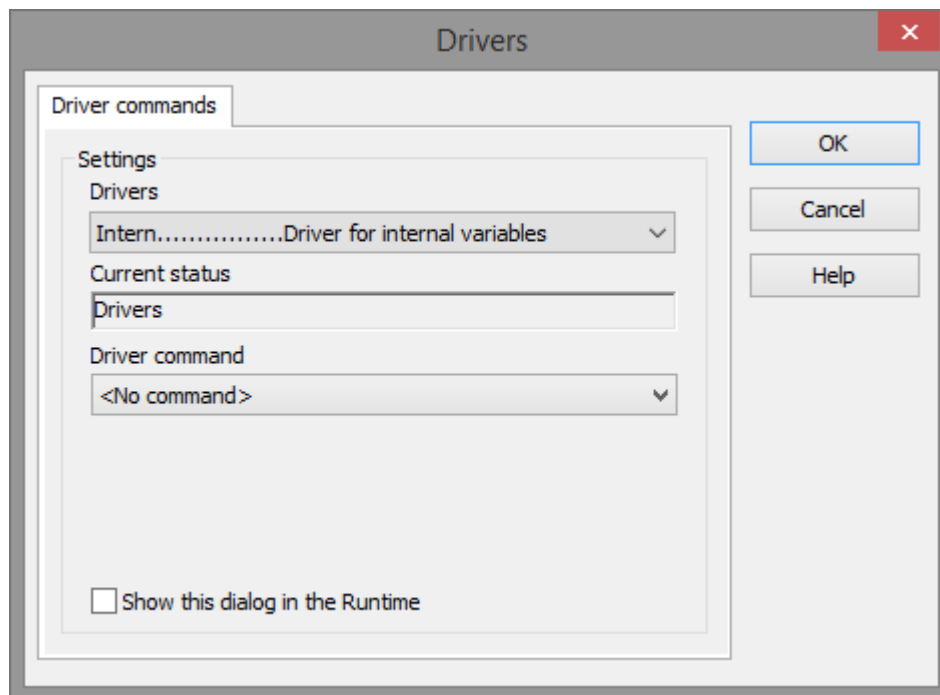
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



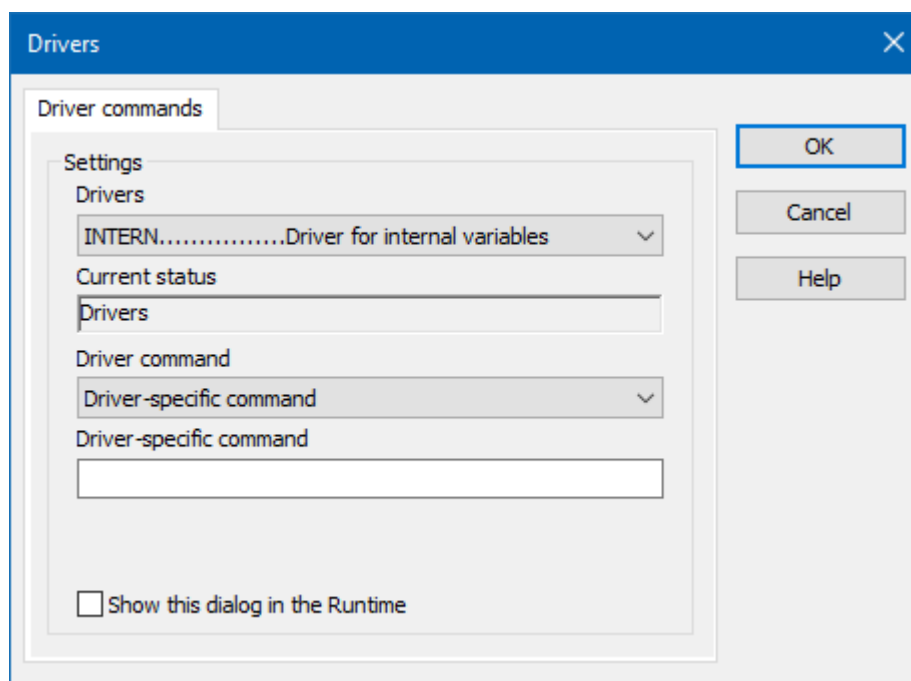
2. Navigate to the node **Variable**.
3. Select the **Driver commands** entry.

The dialog for configuration is opened



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



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 <i>driver-specific command</i> has been selected.
Show this dialog in the Runtime	Configuration of whether the configuration can be changed in the Runtime: <ul style="list-style-type: none"> ▶ <i>Active</i>: This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution. ▶ <i>Inactive</i>: The Editor configuration is applied in the Runtime when executing the function. Default: <i>inactive</i>

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
<i>No command</i>	No command is sent. A command that already exists can thus be removed from a configured function.

Driver command	Description
<i>Start driver (online mode)</i>	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.
<i>Stop driver (offline mode)</i>	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 <i>switched off</i> (OFF; Bit 20).
<i>Driver in simulation mode</i>	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.
<i>Driver in hardware mode</i>	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.
<i>Driver-specific command</i>	Entry of a driver-specific command. Opens input field in order to enter a command.
<i>Driver - activate set setpoint value</i>	Write set value to a driver is possible.
<i>Driver - deactivate set setpoint value</i>	Write set value to a driver is prohibited.
<i>Establish connection with modem</i>	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
<i>Disconnect from modem</i>	Terminate connection (for modem drivers)
<i>Driver in counting simulation mode</i>	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.
<i>Driver in static simulation mode</i>	No communication to the controller is established. All values are initialized with 0.
<i>Driver in programmed simulation mode</i>	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
<i>Communication with driver:<drvExe>/<drvDesc>(id:<drvId>) timed out. No communication for <time> ms.</i>	No communication with driver within the given time. <ul style="list-style-type: none"> ▶ <time>: Time (in milliseconds) ▶ <drvDesc>: Driver name ▶ <drvExe>: Driver EXE name ▶ <drvId>: Driver ID in the zenon project
<i>Communication with %s timed out. Invalid-Bit will be set.</i>	Communication to the %s driver could not be established after 5 attempts within 60 seconds. The <i>INVALID</i> bit is set for the variable.
<i>Communication with %s timed out. Timeout happened %d times</i>	Communication to the %s driver could not be established after %d times within 60 seconds.

10.3 Check list

Check the following in the event of errors:

- ▶ Configuration
- ▶ Connection with external tools
- ▶ Data table number
- ▶ Offset in the data table
- ▶ Driver communication error file
- ▶ Correspondence of building block type (data table type) with the object type

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. You can find more information on the Diagnosis Viewer in the Diagnosis Viewer manual.

The following is required for further analysis of errors:

- ▶ The project backup
- ▶ LOG files

Send these to your support person after agreement with the customer service department.