



zenon
by COPA-DATA

zenon driver manual

MODRTU32

v.8.10



COPADATA

© 2019 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	MODRTU32.....	5
3	MODRTU32 - data sheet	6
4	Driver history.....	7
5	Requirements	8
5.1	PC	8
6	Configuration	8
6.1	Creating a driver.....	9
6.2	Settings in the driver dialog.....	12
6.2.1	General.....	13
6.2.2	Com.....	17
6.2.3	Settings.....	19
6.2.4	Connection TCP/IP	24
7	Creating variables.....	27
7.1	Creating variables in the Editor.....	27
7.2	Addressing	31
7.3	Driver objects and datatypes.....	32
7.3.1	Driver objects	32
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).....	44
8	Driver-specific functions.....	50
9	Driver command function	50
10	Error analysis	55
10.1	Analysis tool	55
10.2	Error numbers.....	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 Support Team, who 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 MODRTU32

The **MODRTU32** driver is a Modbus Master. (Process Gateway is available for Modbus slave).

The following protocols are not supported by the driver:

- ▶ *Modbus ASCII*
for serial connections
- ▶ *Modbus RTU*
for serial and TCP/IP connections

- *Open Modbus TCP*
for TCP/IP connections.

3 MODRTU32 - data sheet

General:	
Driver file name	MODRTU32.exe
Driver name	Modbus RTU und Open Modbus TCP
PLC types	All controllers that support Modbus RTU, Open Modbus TCP or Modbus ASCII (serial)
PLC manufacturer	ABB; Gantner; GE Automation&Controls; Modbus RTU; Mondial; Schiele; Telemecanique; Schneider; Wago; SE Elektronik; Areva; GE Multilin; CIMON

Driver supports:	
Protocol	Modbus RTU - serial; Open Modbus TCP; Modbus RTU over TCP; Modbus ASCII - serial
Addressing: Address-based	Address based
Addressing: Name-based	--
Spontaneous communication	--
Polling communication	X
Online browsing	--
Offline browsing	--
Real-time capable	--
Blockwrite	X
Modem capable	--
RDA numerical	X
RDA String	--
Hysteresis	X

Driver supports:	
extended API	X
Supports status bit WR-SUC	X
alternative IP address	--

Requirements:	
Hardware PC	RS 232 interface or standard network card
Software PC	--
Hardware PLC	--
Software PLC	--
Requires v-dll	X

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

4 Driver history

Date	Driver version	Change
07.07.08	4400	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

In order to establish a serial connection to the PLC, a serial COM port is required at the computer. The port must not be used and therefore blocked by other software (e.g. modem drivers, PLC software, ...) during Runtime.

This driver supports a connection via the standard network card of the PC. Make sure that the PLC and the PC are in the same network range and that the subnet masks are set accordingly on both devices.

6 Configuration

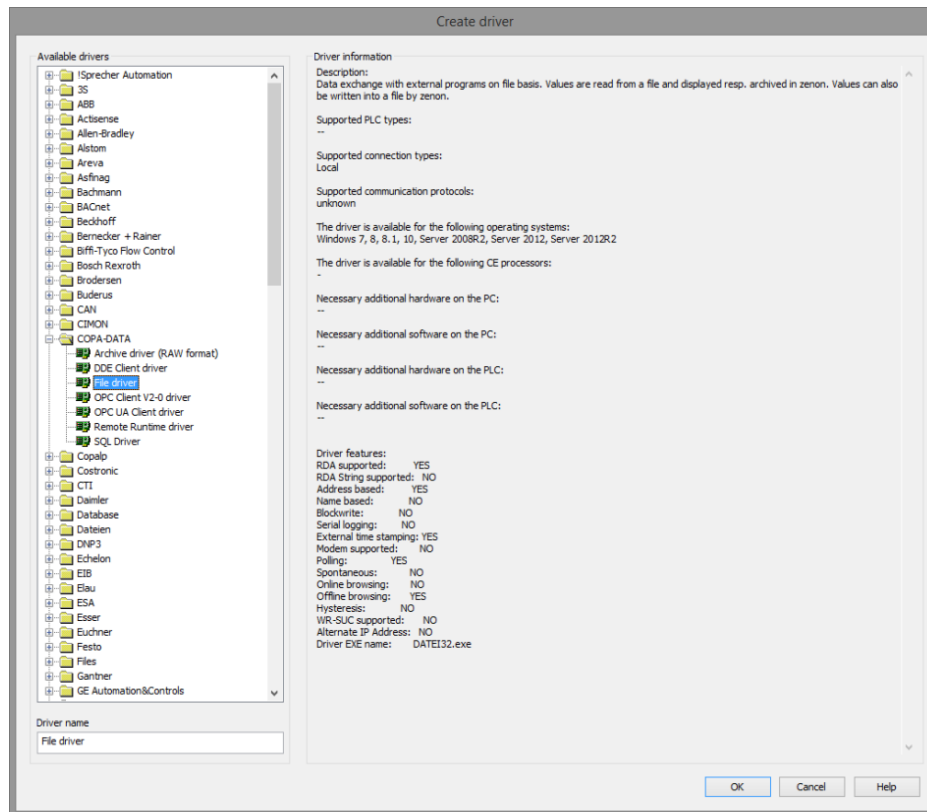
In this chapter you will learn how to use the driver in a project and which settings you can change.

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 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:

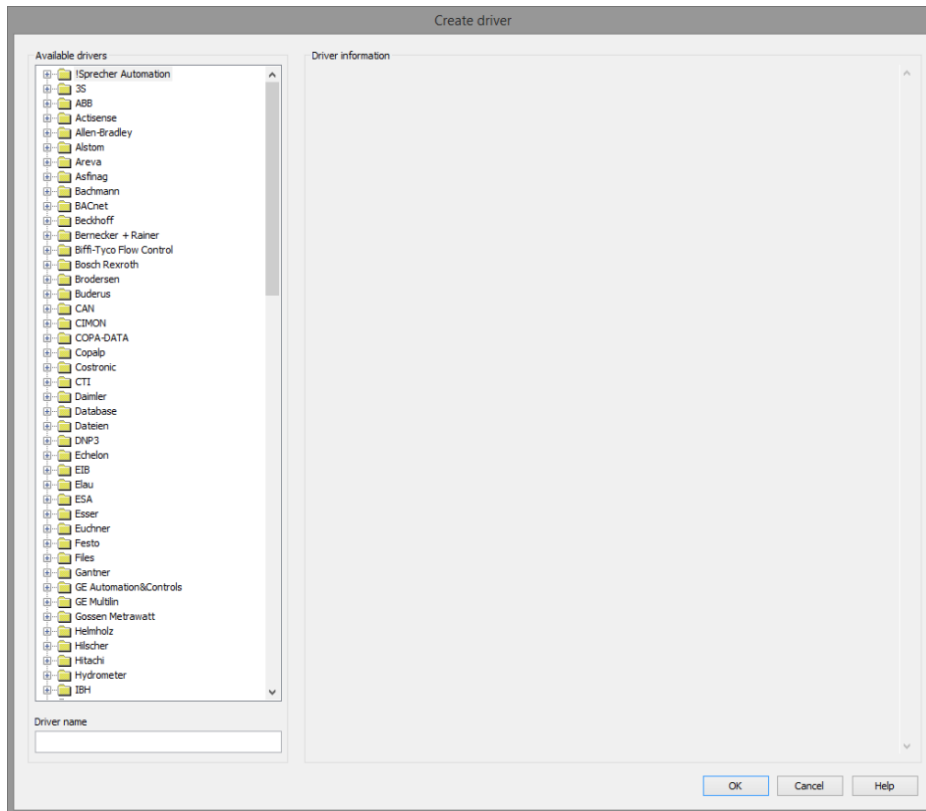
C:\ProgramData\COPA-DATA\zenon[version number].

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 dialog offers a list of all available drivers.

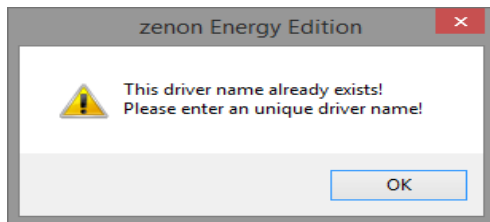


- Select the desired driver and name it in the **Driver name** input field.
This input field corresponds to the **Identification** property. The name of the selected driver is automatically inserted into this input field by default.
The following is applicable for the **Driver name**:
 - ▶ The **Driver 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.
- 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

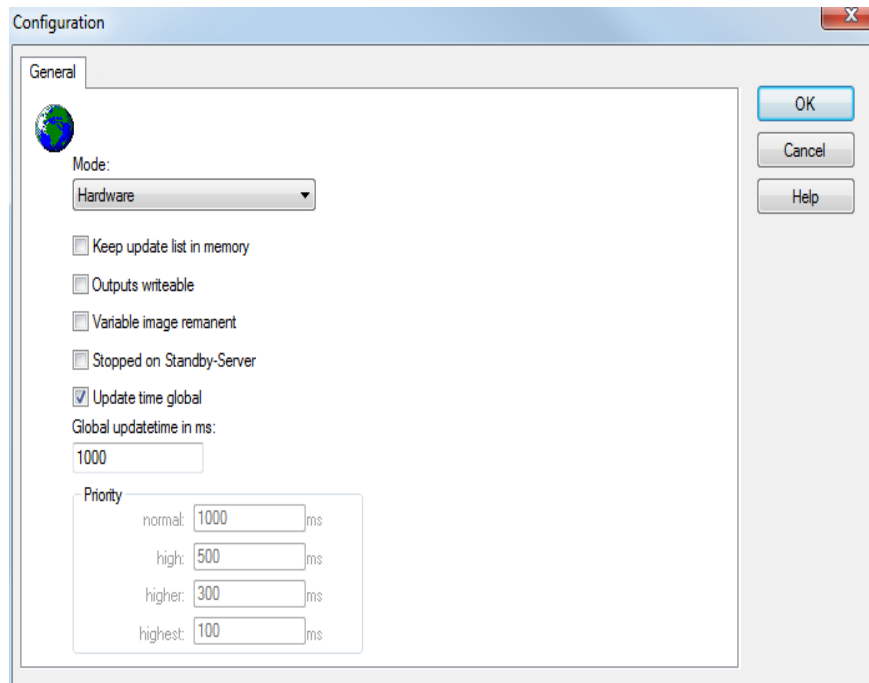
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

6.2 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.



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

Option	Description
	<p>within a value range automatically.</p> <ul style="list-style-type: none"> ▶ <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 (main.chm::/25206.htm).
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. <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 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 object type Driver variable ▶ the driver runs in simulation mode. (not

Option	Description
	<p>programmed simulation)</p> <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 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>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.

Option	Description
	Exceptions: Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the Spontaneous driver update time section.
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.

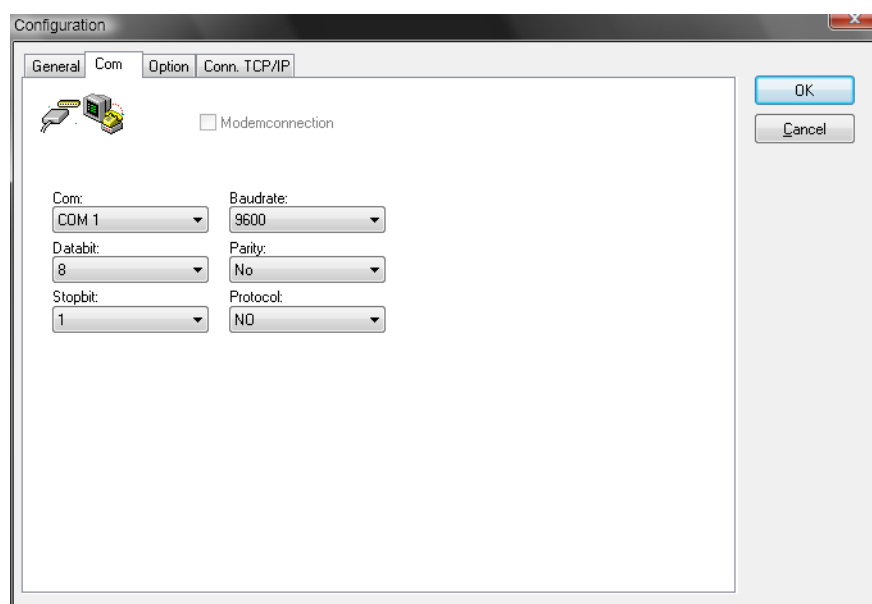
UPDATE TIME FOR SPONTANEOUS DRIVERS

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**.

6.2.2 Com

In this dialog, you configure the connection parameters for the serial communication to the PLC.



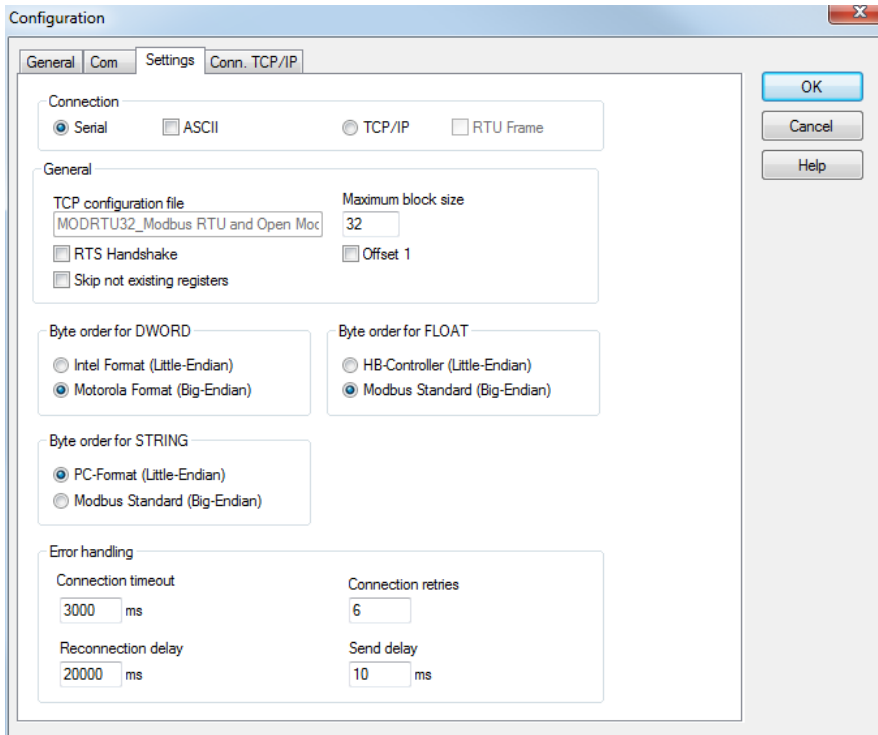
Parameter	Description
Com	Selection of the serial interface (com port) on the computer. Select from drop-down list: <i>COM 1 to COM 256</i> Default: 1
Baud rate	Selection of the Baud rate for the communication to the PLC. The Baud rate must be amended to the PLC. Selection from drop-down list: <i>110 to 1152000</i> Default: 9600
Data bit	Number of data bits (data word length in bits) for communication to the PLC. The data bit rate must be amended to the controller. Select from drop-down list: <i>5, 6, 7, 8</i> Default: 8
Stop bit	Selection of the stop bit for communication to the PLC. The stop bit must be amended to the controller. Select from drop-down list:

Parameter	Description
	<ul style="list-style-type: none"> ▶ 1 ▶ 1.5 ▶ 2 <p>Default: 1</p>
Parity	<p>Selection of the parity for the communication to the PLC. The parity must be amended to the controller. Amend to PLC. Select from drop-down list:</p> <ul style="list-style-type: none"> ▶ No ▶ Odd ▶ Even <p>Default: No</p>
Protocol	<p>Selection of the protocol for communication to the PLC. The protocol must be amended to the controller. Select from drop-down list:</p> <ul style="list-style-type: none"> ▶ No ▶ Rts/CTS <p>Default: No</p>

CLOSE DIALOG

OK	Applies all changes in all tabs and closes the dialog. Only available if no connection is in the <i>edit</i> state.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

6.2.3 Settings



The screenshot shows the 'Configuration' dialog box with the 'Settings' tab selected. The 'Conn. TCP/IP' sub-tab is active. The 'Connection' section has radio buttons for 'Serial' (selected), 'ASCII', 'TCP/IP', and 'RTU Frame'. The 'General' section includes a text field for 'TCP configuration file' (MODRTU32_Modbus RTU and Open Mod), a numeric field for 'Maximum block size' (32), and checkboxes for 'RTS Handshake' and 'Skip not existing registers'. The 'Byte order for DWORD' section has radio buttons for 'Intel Format (Little-Endian)' and 'Motorola Format (Big-Endian)' (selected). The 'Byte order for FLOAT' section has radio buttons for 'HB-Controller (Little-Endian)' and 'Modbus Standard (Big-Endian)' (selected). The 'Byte order for STRING' section has radio buttons for 'PC-Format (Little-Endian)' (selected) and 'Modbus Standard (Big-Endian)'. The 'Error handling' section includes numeric fields for 'Connection timeout' (3000 ms), 'Connection retries' (6), 'Reconnection delay' (20000 ms), and 'Send delay' (10 ms). Buttons for 'OK', 'Cancel', and 'Help' are on the right.

CONNECTION

Parameter	Description
Connection	Definition of which connection is used.
Seriell	For serial mode, both Modbus RTU and Modbus ASCII can be selected.
ASCII	<p>► <i>Active:</i> The driver communicates using the Modbus ASCII protocol.</p> <p><i>Inactive:</i> The driver communicates using Modbus RTU.</p>
TCP/IP	Requires a configuration file. This file stores the TCP/IP settings.
RTU Frame	<i>Active:</i> Serial information is transferred to the TCP protocol exactly.

GENERAL

Parameter	Description
General	General settings.
TCP configuration file	The name of the TCP configuration file is defined here. Default: MODRTU32_[Treiberbezeichnung].txt
Maximum block size	Maximum block size of a data telegram in WORDs. Attention zenon before 5.50 SP6: Block size must be less than 64. (1-125, Default 32)
RTS Handshake	Some PLCs need the RTS handshake for correct communication.
Offset 1	The "Offset 1" affects the addressing of all variables. <i>Active:</i> The driver subtracts 1 when the variable addresses (coils, register) are sent and adds 1 when they are received.
Skip non-existent tabs	The driver reads data in a block. If there is no data between two address areas close to each other, the driver will still try to access these non-existent areas. This option will avoid that.

BYTE SEQUENCE FOR DWORD

Parameter	Description
Byte sequence for DWORD	Defines the sequence of lower-value and higher-value words for double word objects (DINT/UDINT). You can choose between Motorola (Big-Endian) and Intel (Little-Endian).
Motorola Format (Big-Endian)	<i>Active:</i> DWORD ordering in accordance with Motorola format.
Intel Format (Little-Endian)	<i>Active:</i> DWORD ordering in accordance with Intel format.

Note: The sequence of the sorting of low and high values for bytes within a **WORD** always corresponds to the Motorola format.

BYTE SEQUENCE FOR FLOAT

Parameter	Description
Byte sequence for FLOAT	Defines the sequence of lower-value and higher-value words for FLOAT objects (REAL). You can choose between Modbus Standard (Big-Endian) and HB-Controller (Little-Endian).

Parameter	Description
Modbus Standard (Big-Endian)	<i>Active:</i> Float ordering in accordance with Modbus standard.
HB Controller Float (Little-Endian)	<i>Active:</i> Float ordering in accordance with HB PLC.

Note: The sequence of the sorting of low and high values for bytes within a **WORD** always corresponds to the Motorola format.

BYTE SEQUENCE FOR STRING

Parameter	Description
Byte sequence for STRING	Defines display of the byte sequence. Note: If there is relevant information in the documentation of a PLC the following is usually applicable: MSB-first = Motorola = Big-Endian = Modbus. And: Intel = Little-Endian = PC. This can be handled differently in some documentation however.
Modbus Standard (Big-Endian)	<i>Active:</i> Display in accordance with Modbus standard with switched characters.
PC Format (Little-Endian)	<i>Active:</i> Display in PC format with ascending sequence.

ERROR HANDLING

Parameter	Description
Error handling	
Connection timeout	Time in milliseconds to wait for a response from the slave. A communication error will be displayed if there is no response within this time. Only available with serial connection. Note: The communication timeout time must at any rate be higher than the maximum response time of the Modbus RTU Slave. Otherwise misinterpretation of the responses can occur. Default: 3000 ms
Retries on error	Number of send repetitions when there is no answer from the slave after the set communication timeout . 1: a connection attempt, no repetitions

Parameter	Description
	<i>0</i> : constant repetition Default: 6
Delay after connection termination	Time in milliseconds to wait after a communication error has occurred before trying to re-establish the connection. Default: 20000 ms
Send delay	Time delay in milliseconds for "send" orders. Affects the whole driver. Can only be selected with a serial connection.

CLOSE DIALOG

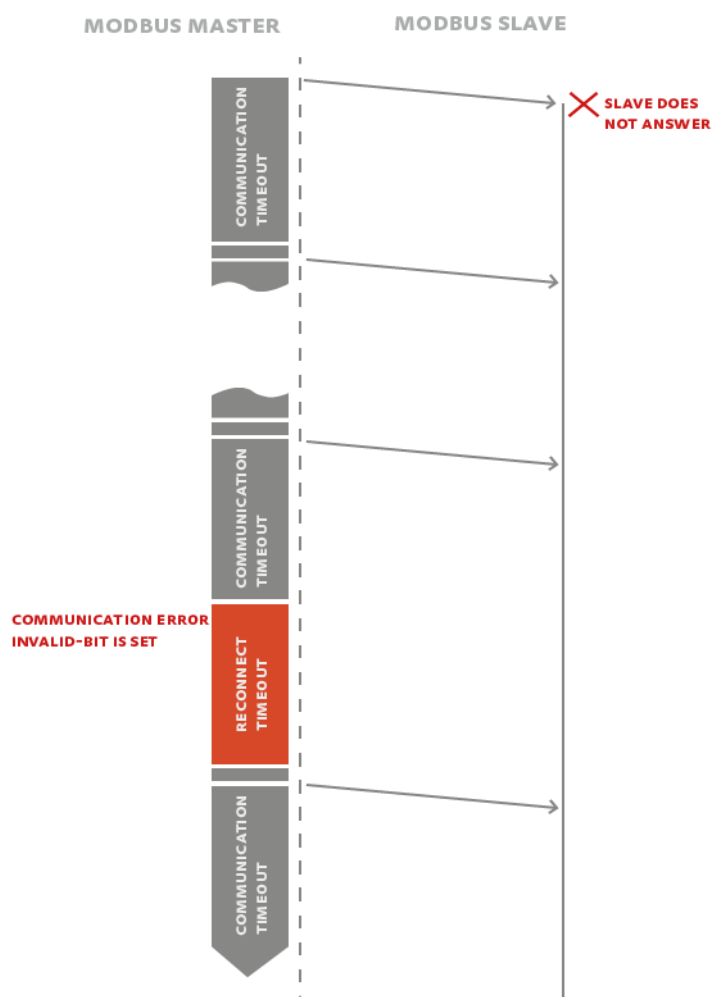
OK	Applies all changes in all tabs and closes the dialog. Only available if no connection is in the <i>edit</i> state.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

TIME OUT BEHAVIOR

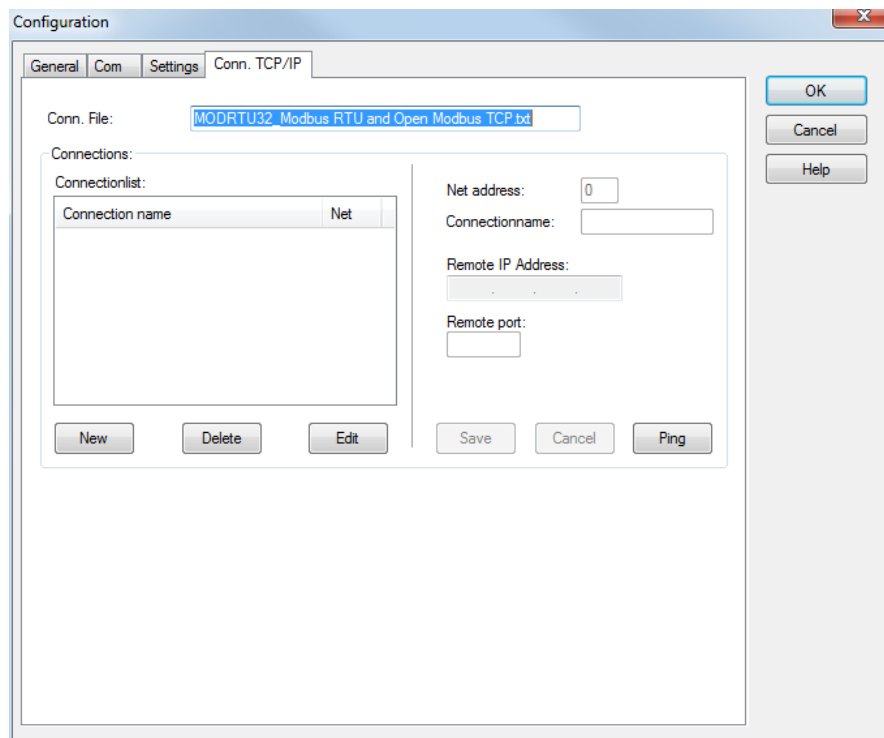
With TCP connections, the communication time out is predefined by the system-dependent property **Timeout [s]**.

After the timeout has expired, the driver attempts - with default settings - to send the request again up to 5 times. If these 5 trials are not successful, no additional attempt to establish a connection is made during the **Reconnect timeout**.

With serial connection the times from the driver configuration are used for the time out behavior. The sent delay time must pass before a request is sent. After a request has been sent, the time period which is waited for an answer is as long as the connection timeout. If there is no answer in this time period, the request is sent again up to 5 times. (a default of 6 means 5 additional attempts). After that no new connection establishment is tried for the set reconnect time out. See also the following graphic.



6.2.4 Connection TCP/IP



The screenshot shows the 'Configuration' dialog box with the 'Conn. TCP/IP' tab selected. The 'Conn. File' field contains 'MODRTU32_Modbus RTU and Open Modbus TCP.txt'. The 'Connections' section has a table with columns 'Connection name' and 'Net'. Below the table are 'New', 'Delete', and 'Edit' buttons. To the right are 'Net address' (0), 'Connectionname' (empty), 'Remote IP Address' (empty), and 'Remote port' (empty). At the bottom are 'Save', 'Cancel', and 'Ping' buttons. On the far right are 'OK', 'Cancel', and 'Help' buttons.

Parameter	Description
Connection file	<p>Name of the file in which the settings for each of the TCP/IP participants are saved.</p> <p>Display only. Changes to the file name are made in the Settings (on page 19) tab.</p>

CONNECTION LIST

Settings of the connections.

Parameter	Description
Connection list	<p>List of defined connections to PLCs.</p> <ul style="list-style-type: none"> ▶ <i>Connection name</i> Connection name, as configured in the connection name. ▶ <i>Net:</i> Net address of the connection, as configured in the Net address property.
New	Creates a new connection and unlocks the input fields in the Connection parameters area.

Parameter	Description
Edit	<p>Unlocks the input fields in the Connection parameters area for a selected connection.</p> <p>Not active if no connection is selected in the connection list.</p>
Delete	<p>Deletes a selected connection from the connection list.</p> <p>Attention: The connection is deleted without requesting confirmation.</p> <p>Not active if no connection is selected in the connection list.</p>

CONNECTION PARAMETERS

You configure the settings of a connection in this area.

The entry is validated. A corresponding warning dialog is shown in the event of an error.

Parameter	Description
Net address	<p>Each connection is assigned a network address. This must correspond to the settings in the Net address property of the variable definition.</p> <p>Default: 1</p> <p>Attention: The value 0 is reserved for broadcasts. The value 255 is reserved for the gateway.</p> <p>Note: For MODBUS TCP communication via a gateway (TCP/IP on RS485) the address of the destination station (the Modbus unit ID or slave address) must be used. With a direct MODBUS TCP connection, the Unit ID address field is ignored by some devices, because the TCP/IP address of the PLC is sufficient in principle. In this case, any desired value can be set.</p>
Connection name	<p>Name of the connection as it is displayed in the connection list.</p> <p>Default: <i>Defaultname</i></p>
Remote IP address	<p><i>Active:</i> The PLC is addressed using the IP address. Entry in the Address input field.</p>
Remote hostname	<p><i>Active:</i> The PLC is addressed using the host name. Entry in the Address input field.</p>

Parameter	Description
Entry of the address	<p>Entry of the address of the PLC. Depending on the settings of the higher-level option field, input is as follows:</p> <ul style="list-style-type: none"> ▶ IP address Default: <i>192.168.0.1</i> ▶ Host name Default: <i>localhost</i>
Remote port	<p>Entry of the port address of the PLC. You can find details in the manual of your PLC.</p> <p>Standard port: <i>502</i></p>
Ping	Sends a ping to the IP address that is configured for this connection. Allows the connection to the device to be tested. If the ping is concluded negatively, check the IP address and check to see if the device is online.
Save	Accepts all changes for edited connection and closes editing option.
Cancel	Discards all changes for edited connection and closes editing option.

CLOSE DIALOG

OK	Applies all changes in all tabs and closes the dialog. Only available if no connection is in the <i>edit</i> state.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

CREATE NEW CONNECTION

1. Click on the **New** button.
2. Enter the connection details.
3. Click on **Save**.

EDIT CONNECTION

1. Select the connection in the connection list.

2. Click on the **Edit** button
3. Change the connection parameters.
4. Finish with **Save**.

DELETE CONNECTION

1. Select the connection in the connection list.
2. Click on the button **Delete**.
3. The connection will be removed from the list

7 Creating variables

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

7.1 Creating variables in the Editor

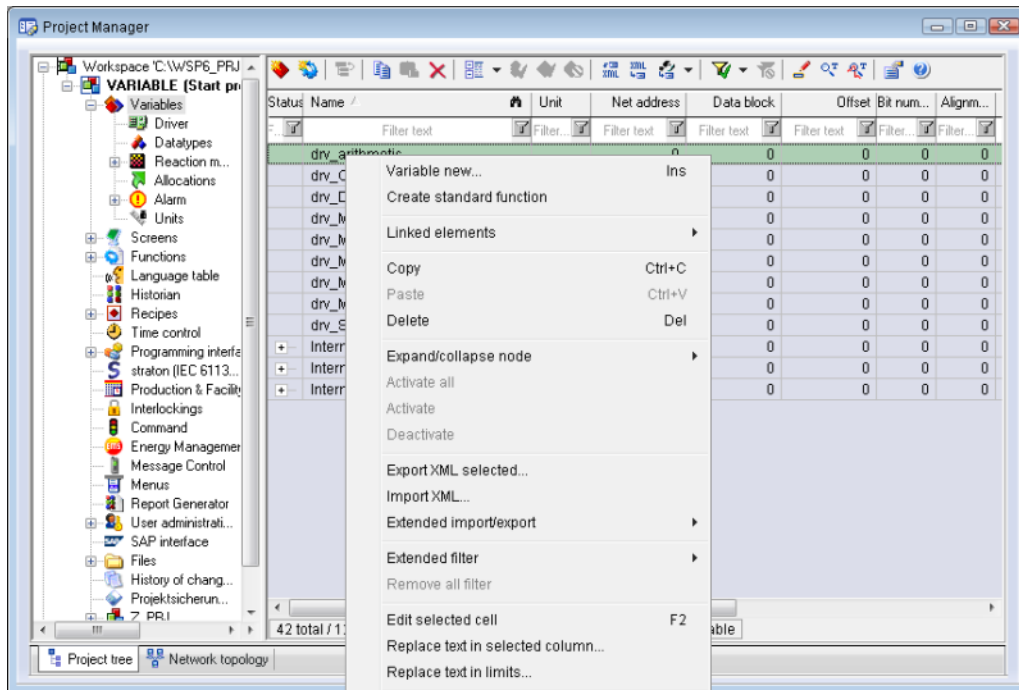
Variables can be created:

- ▶ 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:

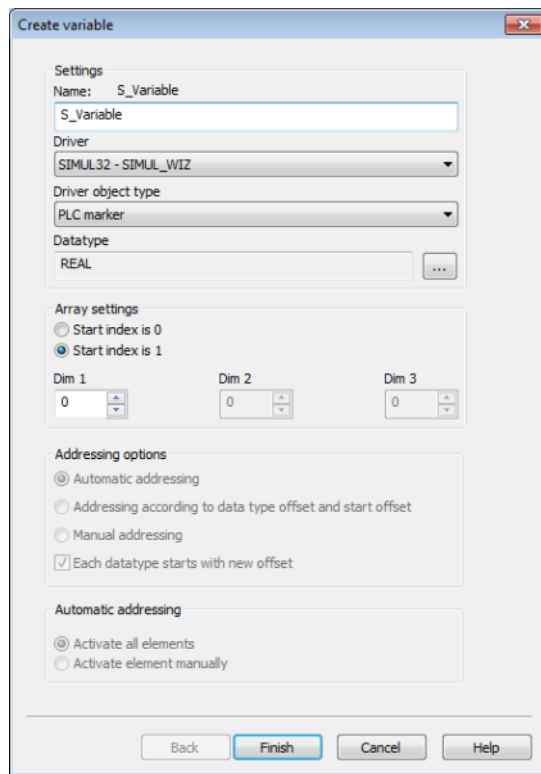
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

2. Configure the variable
3. The settings that are possible depends 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: For some drivers, the addressing is possible over the property Symbolic address, as well.</p>
Drivers	<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

Property	Description
	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 addressing	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
- ▶ KabaDPSEServer
- ▶ OPCUA32
- ▶ Phoenix32
- ▶ POZYTON
- ▶ RemoteRT
- ▶ S7TIA
- ▶ SEL
- ▶ SnmpNg32
- ▶ PA_Drv

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 (Slaveaddress) or net address of the variable.</p> <p>This address relates to serial communication to the Modbus address of the PLC - the address of the recipient.</p> <p>For communication via Ethernet the net address relates to the connection configuration in the driver as well as to the Modbus Unit ID. This defines the PLC/connection on which the variable resides.</p> <p>If a gateway (e.g. TCP/IP on RS485) is used for communicating with the Modbus stations, you have to create a separate TCP/IP connection for each Modbus station. Use the address of the gateway as the IP address. The address of the Modbus station - Modbus Unit ID - is used for the net address of the connection and in the variables.</p> <p>The highest possible Modbus address, according to the protocol specification, is 247.</p> <p>The address 0 is used for Broadcast Messages (write only).</p>
Data block	not used for this driver
Offset	<p>Offset of variables. Equal to the memory address of the variable in the PLC. Adjustable from 0 to 4294967295.</p>
Alignment	<p>The driver uses word-based addressing (16 bit). If only one Byte is read, you can configure here whether the HiByte or the LowByte will be addressed.</p>
Bit number	<p>Number of the bit within the configured offset.</p> <p>Possible entries: 0 to 65535. Working range [0-15]</p>
String length	Only available for String variables.

Property	Description
	Maximum number of characters that the variable can take.
Driver connection/Driver Object Type	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
Driver connection/Data Type	<p>Data type of the variable. Is selected during the creation of the variable; the type can be changed here.</p> <p>Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.</p>
Driver connection/Priority	<p>Setting the priority class. The variable of the priority class is thus assigned as it was configured in the driver dialog in the General tab. The priority classes are only used if the global update time is deactivated.</p> <p>If the global update time option is activated and the priority classes are used, there is an error entry in the log file of the system. The driver uses the highest possible priority.</p>

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: Modbus function (Code hex/dec)	Write: Modbus function (Code hex/dec)	Supported data types	Comment
Alarm Stack	97	0x03/3	N/A	<i>STRING</i>	<p>Read special events from Secheron PLC.</p> <p>Addressing:</p> <ul style="list-style-type: none"> ▶ Event type = data block

Driver-object type	Channel type	Read: Modbus function (Code hex/dec)	Write: Modbus function (Code hex/dec)	Supported data types	Comment
					► Sub-type = offset .
Analog Input	10	0x04/4	N/A	<i>REAL, LREAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING</i>	Class 1 - input register. Linear addressing as with <i>holding register</i> .
Byte aligned Coil	69	0x01/1	0x0F/15	<i>BOOL</i>	Class 1 - coil. Linear addressing one-step via offset .
Coil	65	0x01/1	0x05/5	<i>BOOL</i>	Class 1 - coil. Linear addressing one-step via offset .
Device Status	24	0x11/17	N/A	<i>BOOL, USINT</i>	PLC specific.
Device Identification	68	0x2B/43	N/A	<i>STRING</i>	PLC specific.
Input Status	66	0x02/2	N/A	<i>BOOL</i>	Class 1 - input discretes. Linear addressing one-step via offset .
Holding Register	8	0x03/3	0x10/16	<i>REAL, LREAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING, WSTRING</i>	Class 0 - multiple register. Linear addressing: <ul style="list-style-type: none"> ► Bool (1 Bit): one-step via Offset and bit number ► Byte (8 bits): one-step via Offset and

Driver-object type	Channel type	Read: Modbus function (Code hex/dec)	Write: Modbus function (Code hex/dec)	Supported data types	Comment
					Orientation <ul style="list-style-type: none"> Word (16 bits) Double word (32 bits) Float (32 bits) String(n*Byte): one-step via Offset WString (n*Word): one-step via Offset
Holding Register 32 Bit	70	0x03/3	0x10/16	<i>REAL</i>	32 bits (4 bytes) are read per offset.
Holding Register 64 Bit	71	0x03/3	0x10/16	<i>LREAL</i>	64 bits (8 bytes) are read per offset.
Preset Single Register (FC 6)	67	N/A	0x06/6	<i>UINT</i>	Class 1 - single register.
<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; is 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</p>

Driver-object type	Channel type	Read: Modbus function (Code hex/dec)	Write: Modbus function (Code hex/dec)	Supported data types	Comment
					information on this in the Communication details (Driver variables) (on page 44) chapter.

Key:

X: supported

--: not supported

MODBUS FUNCTION CODES

Function code hex/dec	Modbus identifier	Comment
0x01 / 1	Read coils	This function code is used to read from 1 to 2000 contiguous status of coils (bits) in a remote device
0x02/2	Read discrete inputs	This function code is used to read from 1 to 2000 contiguous status of discrete inputs (bits) in a remote device
0x03 / 3	Read multiple registers	This function code is used to read a block of contiguous holding registers (1 to 125 words) in a remote device.
0x04 / 4	Read input registers	This function code is used to read a block of contiguous input registers (1 to 125 words) in a remote device.
0x05 / 5	Write coil	This function code is used to write a single output (one bit) to either ON or OFF in a remote device.
0x06 / 6	Write single register	This function code is used to write a single (one word) holding register in a remote device.
0x10 / 16	Write multiple registers	This function code is used to write a block of contiguous holding registers (1 to approx. 120 words) in a remote device.

Function code hex/dec	Modbus identifier	Comment
0x11 / 17	Report Slave ID	This function code is used to read the description of the type, the current status, and other information specific to a remote device.
0x14/20	Read File Record	<p>This function code is used to perform a file record read.</p> <p>All Request Data Lengths are provided in terms of number of bytes and all Record Lengths are provided in terms of registers.</p> <p>Note: This Functioncode is supported by the MODBUS_ENERGY driver only.</p>
0x15/21	Write File Record	<p>This function code is used to perform a file record write.</p> <p>All Request Data Lengths are provided in terms of number of bytes and all Record Lengths are provided in terms of the number of 16-bit words.</p> <p>Note: This Functioncode is supported by the MODBUS_ENERGY driver only.</p>
0x2B/43	Read Device Identification	This function code allows reading the identification and additional information relative to the physical and functional description of a remote device.

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.

EXAMPLE FOR ALL POSSIBLE ZENON DATA TYPES:

PLC	zenon	Data type
BOOL	BOOL	8
BYTE	SINT	10
BYTE	USINT	9

PLC	zenon	Data type
WORD	INT	1
WORD	UINT	2
DWORD	DINT	3
DWORD	UDINT	4
FLOAT	REAL	5
FLOAT	LREAL	6
STRING	STRING	12
WSTRING	WSTRING	21

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 (main.chm::/13028.htm) manual in the Variables (main.chm::/13045.htm) 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 (main.chm::/13046.htm)** 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 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 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 the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered manually). The length can be limited using the MAX_LAENGE entry in the project.ini file.
KANAL_D	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the MAX_LAENGE entry in the project.ini file.
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.

Identification	Type	Field size	Comment
HWKANAL	Num	3	Net address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager
LES_SCHR	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-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)

Identification	Type	Field size	Comment
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 VALUE DEFINITION

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

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

Identification	Type	Field size	Comment
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
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

Name from import	Type	Offset	Description
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 status of Runtime for driver simulation.
ConnectionStates	STRING	61	Internal connection status of the driver to the PLC. Connection statuses: 0: Connection OK 1: Connection failure 2: Connection simulated Formating: <Netzadresse>:<Verbindungszustand>;...;...; A connection is only known after a variable has first signed in. In order for a connection to be contained in a string, a variable of this connection must be signed in once. The status of a connection is only updated if a variable of the connection is signed in. Otherwise there is no communication with the corresponding controller.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	BOOL	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).

Name from import	Type	Offset	Description
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
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

Name from import	Type	Offset	Description
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.
MinBlkReadTime	<i>UDINT</i>	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	<i>UDINT</i>	33	Number of writing errors
ReadSucceedCount	<i>UDINT</i>	35	Number of successful reading attempts
MaxCycleTime	<i>UDINT</i>	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	<i>UDINT</i>	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	<i>UDINT</i>	26	Number of writing attempts
ReadErrorCount	<i>UDINT</i>	34	Number of reading errors
MaxUpdateTimeNormal	<i>UDINT</i>	56	Time since the last update of the priority group Normal in milliseconds (ms).

Name from import	Type	Offset	Description
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:

BLOCKWRITE

For more efficient writing of set values (such as recipes) of variables of the *Holding Register* driver object type, the Blockwrite property can be activated. Variables that are consecutive in the control memory are thus described with a write telegram. With larger areas, the writing is combined into a few telegrams instead of describing each variable individually.

ACTIVATING BLOCKWRITE

The following entry must be added in the **project.ini**:

[MODRTU32]

BLOCKWRITE=1

TRANSACTION IDENTIFICATION

The driver increases the Transaction Identifier for each packet sent and expects a response from the slave with the same Transaction Identifier or 0.

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

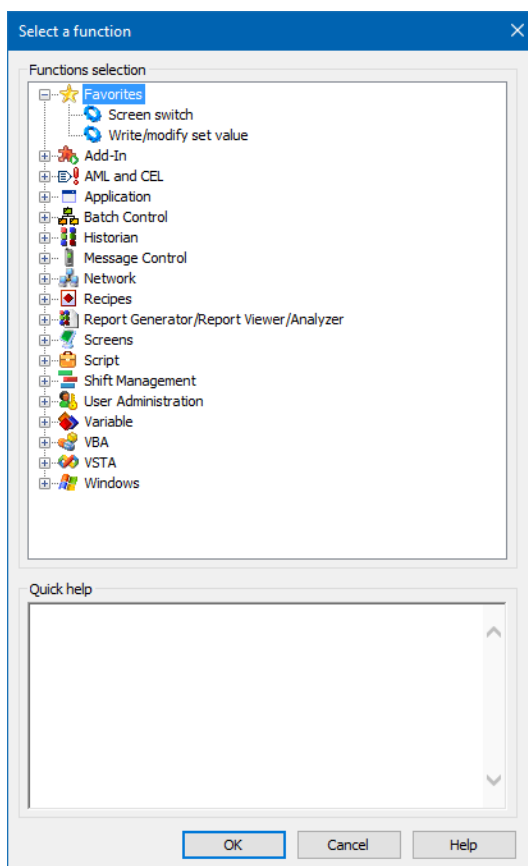
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

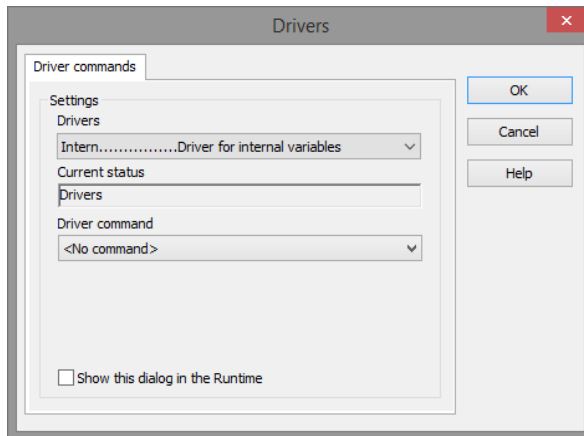
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



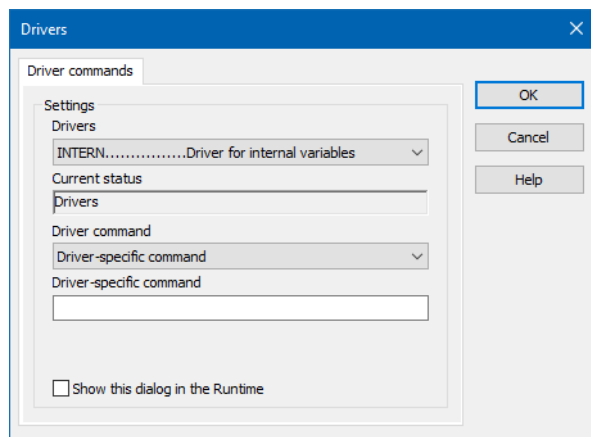
2. Navigate to the node **Variable**.
3. Select the **Driver commands** entry.

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. Has no function in the current version.
Driver command	Selection of the desired driver command from a drop-down list. 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.

Option	Description
	Note: Only available if, for the driver command option, the <i>driver-specific command</i> has been selected.
Show this dialog in the Runtime	<p>Configuration of whether the configuration can be changed in the Runtime:</p> <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. <p>Default: <i>inactive</i></p>

CLOSE DIALOG

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

AVAILABLE DRIVER COMMANDS

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

Driver command	Description
<No command>	<p>No command is sent.</p> <p>A command that already exists can thus be removed from a configured function.</p>
<i>Start driver (online mode)</i>	<p>Driver is reinitialized and started.</p> <p>Note: If the driver has already been started, it must be stopped. Only then can the driver be re-initialized and started.</p>
<i>Stop driver (offline mode)</i>	<p>Driver is stopped. No new data is accepted.</p> <p>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).</p>

Driver command	Description
<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>Activate driver write set value</i>	Write set value to a driver is possible.
<i>Deactivate driver write set 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 (main.chm::/12464.htm) program that was also installed with zenon. You can find it under **Start/All programs/zenon/Tools 8.10 -> 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 (main.chm::/12464.htm) manual.

10.2 Error numbers

The following error codes apply ONLY for the error messages

CallSPSread FAILED ! ERROR:' and

'CallSPSwrite FAILED ! ERROR:'

-1	Invalid function
-2	Invalid MW address
-3	Invalid date
-4	Slave error
-7	Slave cannot execute function
-8	Slave memory parity error
-10	Timeout while communicating with Slave
-11	Read request failed! Communication with PLC impossible.

The fields Para1 and Para2 contain additional information about the variable that cannot be read:

The number relate to the internal ID

Para1: Channel type (e.g.: 8=Holding Register)

8: Holding Register

10 Input Register

24 Status

65 COIL

66 Input Status

67 Preset Single Register

Para2: Data type (e.g: 2= Word, UINT)

1: INT

2: UINT

3: DINT

4: UDINT

5: REAL

8: BOOL

9: SINT

10: USINT

10.3 Check list

Problem	Diagnostics	Reason
Values can be read or written by the controller.	The controller can be contacted by 'pinging'?	<ul style="list-style-type: none"> ▶ The controller is not connected to the power supply or the network. ▶ The PC is not connected to the network. ▶ The controller is connected but is in a different subnetwork and the network gateway is not entered in the controller or the subnetmask is not set correctly. ▶ Is the firewall activated? Port 502 is used for communication as standard; add it to the exceptions. Enter accordingly for individual port numbers.
	The controller can be contacted by 'pinging'?	<ul style="list-style-type: none"> ▶ The communication parameters are not set correctly? ▶ The port must be set according to the configuration of the controller (Modbus Slave). ▶ The network address in the addressing of the variable does not correspond to the

Problem	Diagnostics	Reason
		<p>network address of the connection in the driver. Attention: Network address 0 must not be used. Address 0 is reserved as a broadcast address in Modbus.</p> <ul style="list-style-type: none"> ▶ The driver configuration file was not transferred to the target computer?
	The PLC is connected serially	<ul style="list-style-type: none"> ▶ The controller is not connected to the power supply or the bus system. ▶ The serial cable is not connected to the correct interface (COM1...64), or the interface was set incorrectly. ▶ The serial interface is blocked by another application. ▶ The network address in the addressing of the variable does not correspond to the Modbus address (device address). ▶ The cable is assigned incorrectly or defective.
Certain values cannot be read or written by the controller.	<p>Has an analysis with the Diagnosis Viewer been carried out to see which errors have occurred?</p> <p>See Analysis tool (on page 55) chapter.</p>	<ul style="list-style-type: none"> ▶ See the following chapter: Error numbers (on page 56) ▶ Are the variables correctly addressed? ▶ Are the correct object types used in the variable? The object types determine the function code to be used in the Modbus telegram.
Incorrect values are displayed.	<p>Has an analysis with the Diagnosis Viewer been carried out to see which errors have occurred?</p> <p>See Analysis tool (on page 55) chapter.</p>	<ul style="list-style-type: none"> ▶ Are the variables correctly addressed? ▶ Are the correct data types used? ▶ Is the value calculation correct?