



zenon
by COPA-DATA

zenon driver manual

MtoM32

v.8.10



COPA-DATA

© 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	MtoM32	5
3	MtoM32 - data sheet.....	6
4	Driver history.....	7
5	Configuration	7
5.1	Creating a driver.....	8
5.2	Settings in the driver dialog.....	11
5.2.1	General.....	12
5.2.2	Com.....	16
5.2.3	MtoM settings.....	18
6	Creating variables.....	32
6.1	Creating variables in the Editor.....	32
6.2	Addressing	36
6.2.1	Notes on the system memory of the PMI/PC	37
6.3	Driver objects and datatypes.....	38
6.3.1	Driver objects	38
6.3.2	Mapping of the data types	38
6.4	Creating variables by importing.....	39
6.4.1	XML import	40
6.4.2	DBF Import/Export	41
6.5	Communication details (Driver variables).....	46
7	Driver-specific functions.....	52
8	Driver command function	52
9	Connection cable.....	57
9.1	Connection cable 1 (RS 232C)	57
9.2	Connection cable 2 (RS 422).....	58
9.3	Connection cable 3 (RS 485).....	58
10	Error analysis	59
10.1	Analysis tool	59

10.2 Check list	60
-----------------------	----

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 MtoM32

The **MtoM32** driver is a passive driver with an open protocol for zenon.

All write and read access is controlled by the connected communication partner/host (PLC or PC). The externally-controlled monitor display of the control system only changes after a data transfer from the partner to zenon. The data transfer from zenon to the master is carried out by a read command. The partner must send this read command.

System memory is available as a data buffer in zenon for communication.

Connection is not possible on PC systems with the "RS 485" setting.

3 MtoM32 - data sheet

General:	
Driver file name	MtoM32.exe
Driver name	Memory to Memory Treiber
PLC types	Any Protocol must be implemented.
PLC manufacturer	Pilz

Driver supports:	
Protocol	Pilz MtoM
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	--
Supports status bit WR-SUC	--

Driver supports:	
alternative IP address	--

Requirements:	
Hardware PC	--
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	Build number	Change
20.09.2014	14584	Created new driver documentation.

5 Configuration

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

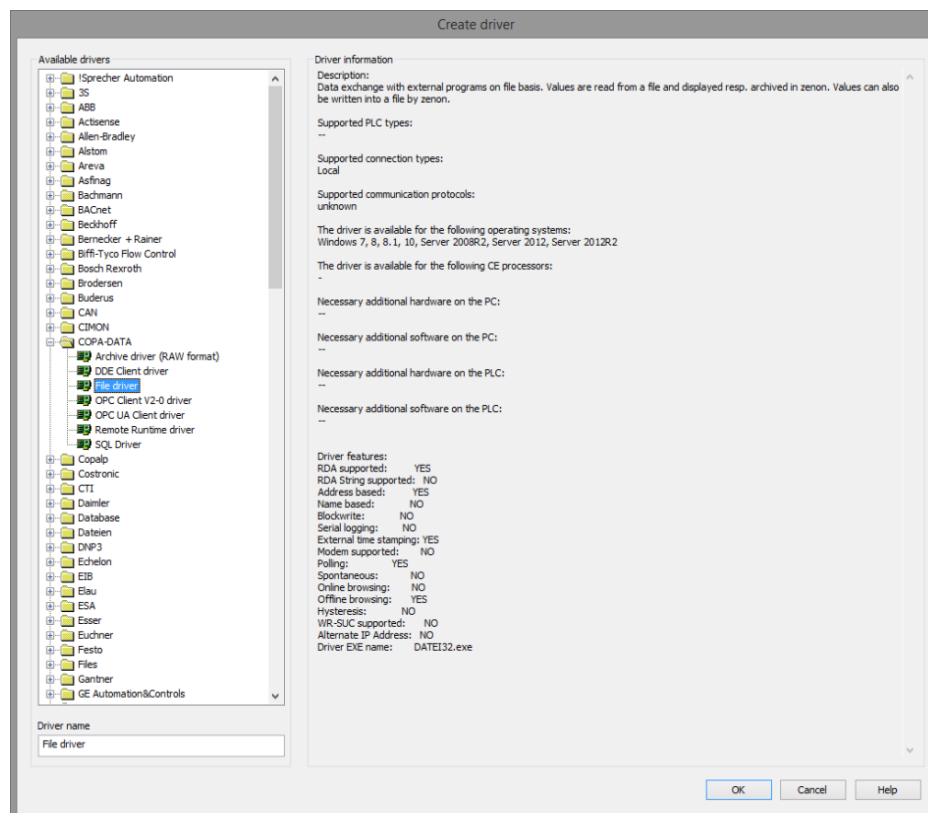


Information

Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.

5.1 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>

Parameter	Description
	Default: <i>Empty</i> The information on the selected driver is shown in this area after selecting a driver.

CLOSE DIALOG

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



Information

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

CREATE NEW DRIVER

In order to create a new driver:

1. Right-click on **Driver** in the Project Manager and select **New driver** in the context menu.
Optional: Select the **New driver** button from the toolbar of the detail view of the **Variables**.
The **Create driver** dialog is opened.

- The 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

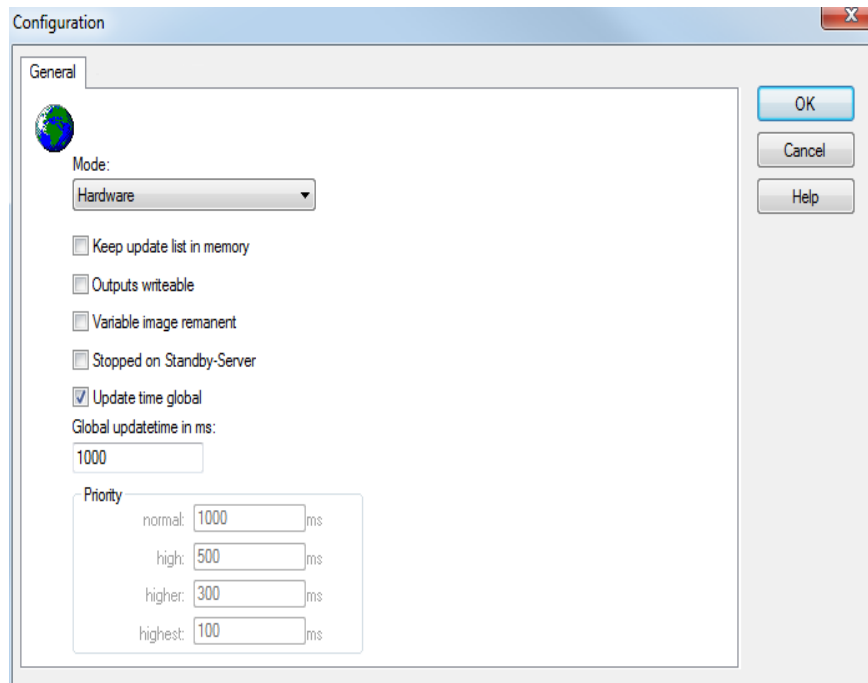
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

5.2 Settings in the driver dialog

You can change the following settings of the driver:

5.2.1 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) to 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

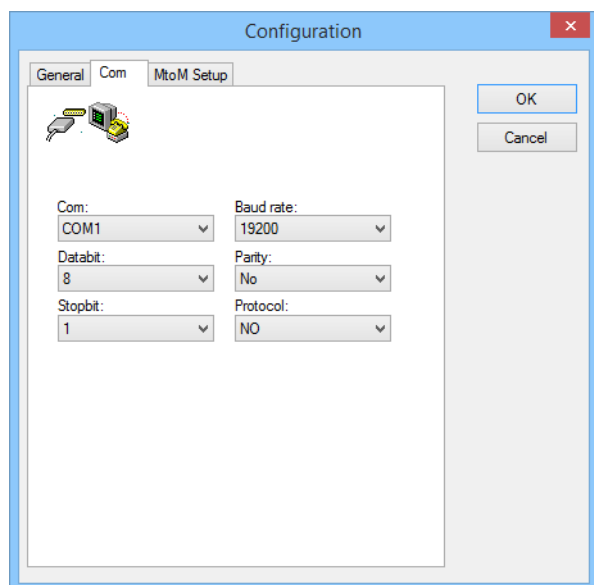
Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

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**.

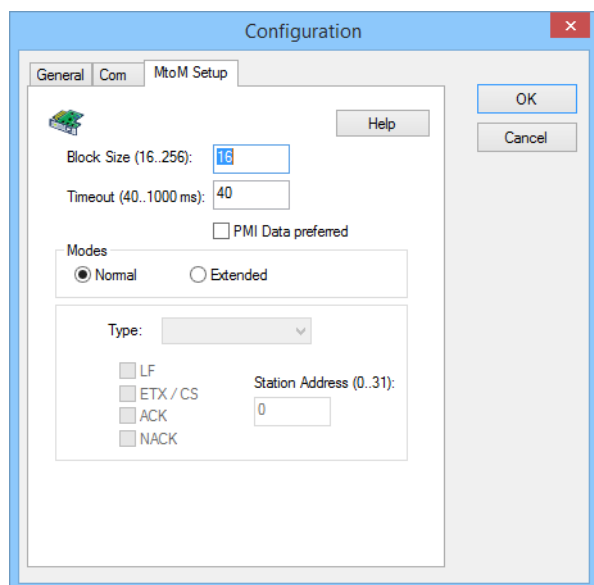
5.2.2 Com



Parameter	Description
Com:	<p>Selection of the serial interface (physical communication port) on the computer via which the communication is carried out: COM 1 to COM 9</p> <p>Default: <i>COM1</i></p>
Data bit:	<p>Variable for selecting the number of data bits:</p> <ul style="list-style-type: none"> ▶ 5 ▶ 6 ▶ 7 ▶ 8 <p>Default: 8</p>
Stop bit:	<p>Drop-down list to select the number of stop bits. Allows the recipient to synchronize to each transferred character.</p> <ul style="list-style-type: none"> ▶ 1 ▶ 1.5 ▶ 2

Parameter	Description
	Default: 1
Baud rate:	<p>Drop-down list to select the Baud rate for interface communication.</p> <p>Selection options: 4800, 9600, 19200, 38400, 56000, 57600, 115200</p> <p>Default: 19200</p>
Parity:	<p>Selection of the parity protocol from a drop-down list:</p> <ul style="list-style-type: none"> ▶ No No parity - no control bit ▶ Odd Parity bit is set if the number of data bits is even. ▶ Even Parity bit is set if the number of data bits is not even. <p>Default: No</p>
Protocol:	<p>Selection of the interface standard for the interface from a drop-down list:</p> <ul style="list-style-type: none"> ▶ NO ▶ RS485 (for increased stability) <p>Default: NO</p>

5.2.3 MtoM settings



Parameter	Description
Block size (16..256)	<p>Internal block size between Runtime and MtoM32 driver.</p> <p>Value range: 16 to 256 (words)</p> <p>Default: 16</p>
Timeout (40...1000 ms)	<p>Character delay time. Timeout time in milliseconds</p> <p>If there are pauses that are longer than the setting within a telegram, the corresponding telegram is discarded.</p> <p>Value range: 40 to 1000</p> <p>Default: 40</p>
PMI data preferred	<p>Multicast communication has priority if activated.</p> <p>Default: inactive</p>
Modes	<p>Drop-down list for the selection of the communication mode.</p> <p>This setting influences the further configuration possibilities for the communication type and the station address.</p>

Parameter	Description
	<ul style="list-style-type: none"> ▶ Normal ▶ Extended <p>Default: normal</p> <p>Also note the notes on system memory (on page 37)</p>
Type	<p>Drop-down list for the selection of the xx type:</p> <ul style="list-style-type: none"> ▶ 1: 1 ASCII ▶ 1:1 Binary ▶ 1:N ASCII ▶ 1:N Binary <p>Only active if mode is Extended</p> <p>Also note the notes on system memory (on page 37)</p>
Additional telegrams	<p>The following functions can be selected in Extended mode:</p> <ul style="list-style-type: none"> ▶ LF Line Feed Note: only active if type is ASCII ▶ ETX / CS End of Text/Checksum ▶ ACK Acknowledge telegram (ACK) ▶ NACK Not-Acknowledge telegram (NAK) <p>The selected functions are appended to the response or send telegram</p>
Station address (0...31):	<p>With the <i>1:n modes</i>, several stations (slave) can be connected to the host (master).</p> <p>The entry of a station address is absolutely necessary in this case.</p> <p>Value range: 0 to 31 (00_{Dec} to 1F_{Hex})</p> <p>Default: 0</p>

Parameter	Description
	<p>Only active if type is 1:N.</p> <p>Note: If 255_{Dec} (FF_{Hex}) is given as the station address, the telegram is received by all control systems. If 255_{Dec} (FF_{Hex}) is given as the station number, the telegram is received from all PMI.</p>



Attention

A response telegram never follows a telegram with the address 255_{Dec} (FF_{Hex}) - not even ACK or NAK.

5.2.3.1 Communication modes

The **MtoM32** driver provides a total of five different communication modes:

- ▶ Normal mode (ASCII codes)
- ▶ Extended mode
 - ▶ 1:1 ASCII mode
 - ▶ 1:1 Binary mode
 - ▶ 1:n ASCII mode
 - ▶ 1:n Binary mode

The *normal mode* and the *Extended mode 1:1 ASCII* or *1:1 Binary* are suitable for point-to-point connections. In doing so, a master (PLC, microcontroller, PC) and a control system/PC are present. The master sends read and write telegrams to the control system/PC.

Extended mode 1:n ASCII or *1:n Binary* allow the connection of several control systems (slaves) to a master. The master then communicates alternately with the connected devices. The sequence of when communication takes place with which slave is determined by the user program in the master. There is also the possibility of a transmission telegram to all slaves (group call), however only with a write command. Therefore data that needs all slaves can be transferred very quickly. With a group call, the connected slaves do not respond. There is also no return information about whether the telegram has been understood or not.

DIFFERENCE BETWEEN ASCII CODE AND BINARY CODE

With the modes *1:n ASCII*, *1:1 ASCII* and *1:n ASCII*, all characters are converted to ASCII characters. The control characters are organized in hex (binary) code. As a result, the data in the telegram is clearly differentiated from the control character.

The binary modes (*1:1 Binary* and *1:n Binary*) transfer the control characters and the data in binary (hex) code. A type of data compression occurs as a result.



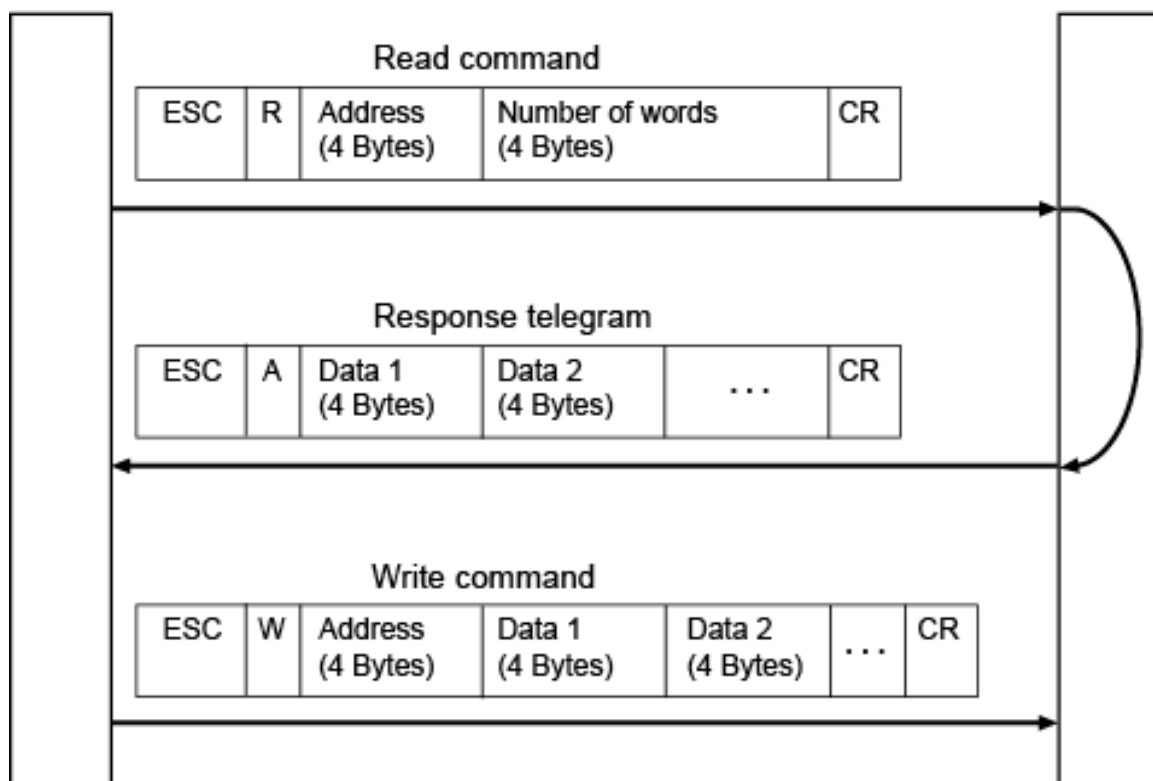
Information

In binary mode, control characters that look like end identifications ETX, CR, LF) can also be present. The master thus has no unique end character for a telegram

5.2.3.1.1 Normal mode

Communications partner

PMI/PC



Character	Identification	Meaning	Hex
<i>R</i>	<i>Read</i>	Reading of data from the PMI/PC	52
<i>A</i>	<i>Answer</i>	Response to read telegram	41
<i>W</i>	<i>Write</i>	Writing of data in PMI/PC	57
<i>CR</i>	<i>Carriage Return</i>	Command end character	0D
<i>ESC</i>	<i>Escape</i>	Command start character	1b

WRITE COMMAND

With a write command, data is transferred from the PLC to the slave (PMI/PC). The start code, the write command and the return are entered in hexadecimal. The address and the data are entered in ASCII code.

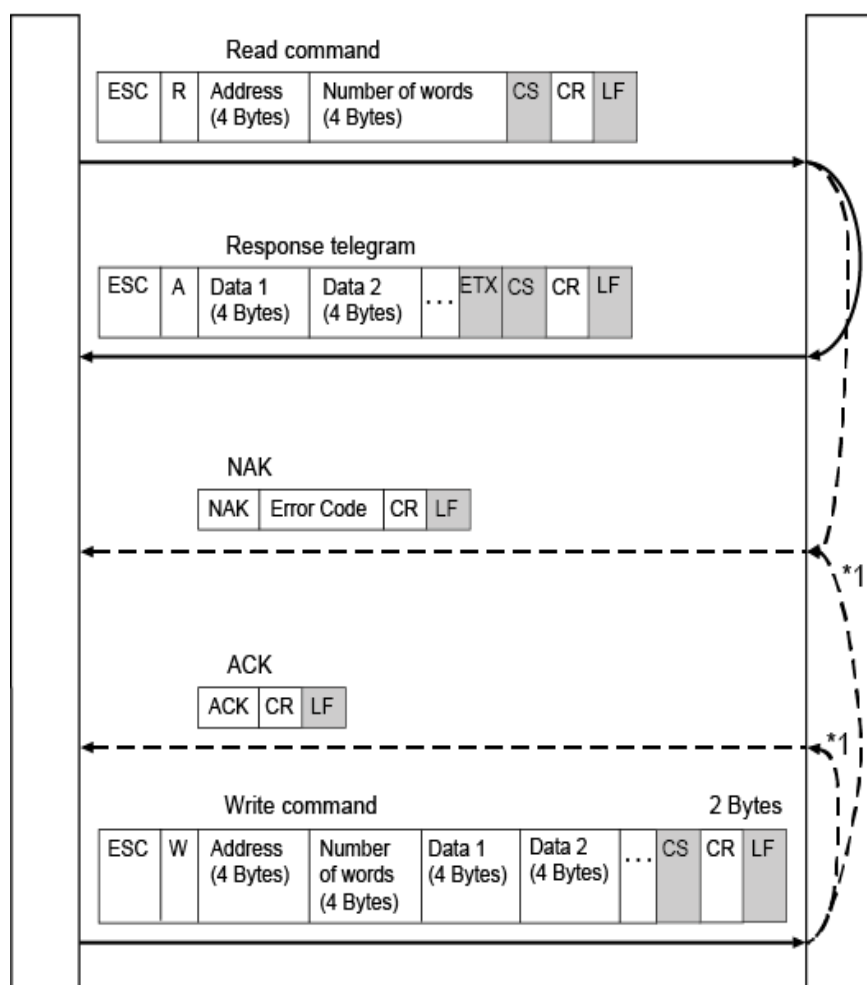
5.2.3.1.2 Extended mode

Optional control characters are underlined in gray in the following illustrations. Optional telegrams are marked with dotted arrow lines

5.2.3.1.31:1 ASCII mode

Communications partner

PMI/PC



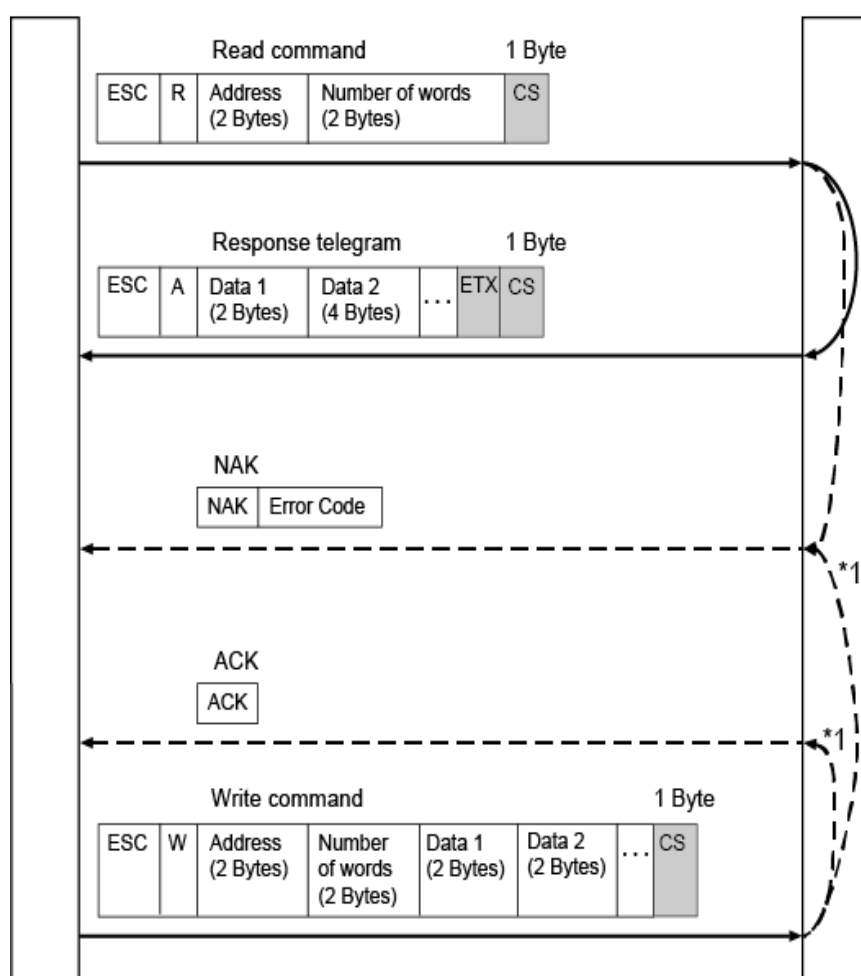
Character	Identification	Meaning	Hex
R	Read	Reading of data from the PMI/PC	52
A	Answer	Response to read telegram	41
W	Write	Writing of data in PMI/PC	57
ACK	Acknowledge	Receive data without errors	06
NAK	Negativ Acknowledge	Errors on receipt of data	15
CR	Carrige Return	End character command	0D

Character	Identification	Meaning	Hex
LF	Line Feed	End character command	0A
ETX	End of Text	End character telegram	03
ESC	Escape	Start character telegram	1B
CS	Checksum	Checksum	-

5.2.3.1.41:1 Binary mode

Communications partner

PMI/PC

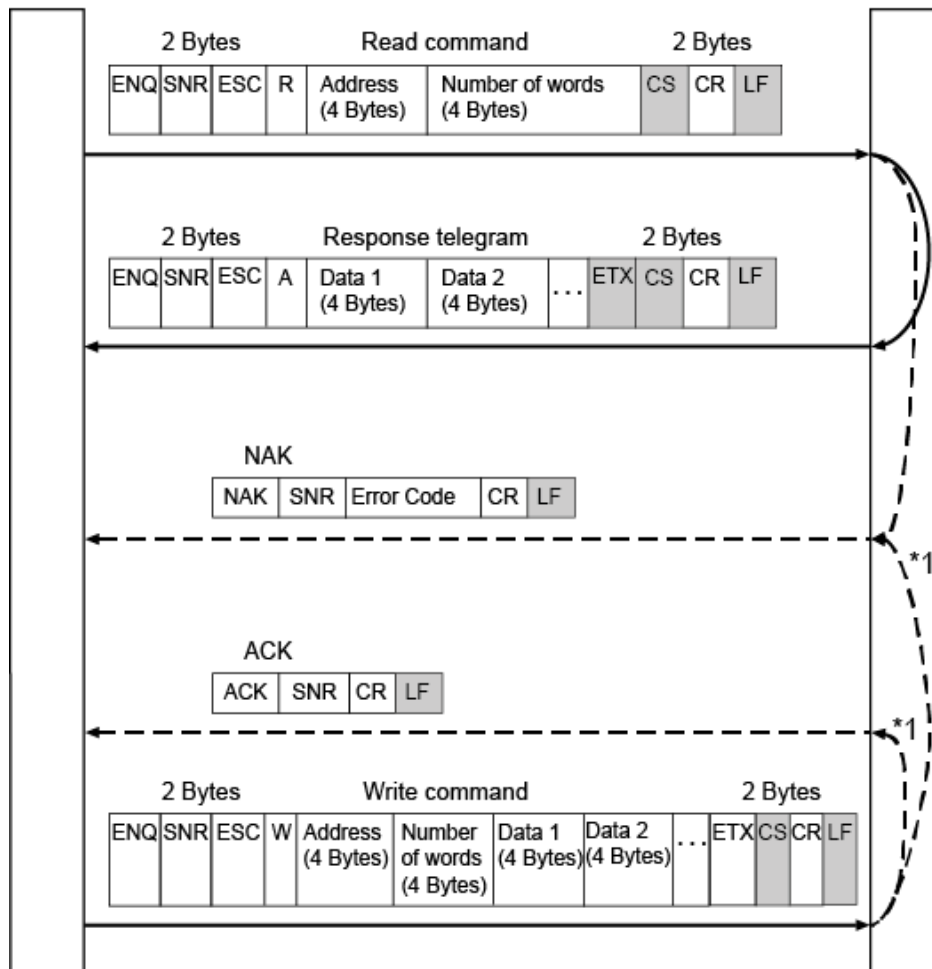


Character	Identification	Meaning	Hex
R	Read	Reading of data from the PMI/PC	52
A	Answer	Response to read telegram	41
W	Write	Writing of data in PMI/PC	57
ACK	Acknowledge	Receive data without errors	06
NAK	Negativ Acknowledge	Errors on receipt of data	15
ETX	End of Text	End character telegram	03
ESC	Escape	Command start character	1B
CS	Checksum	Checksum	-

5.2.3.1.51:n ASCII mode

Communications partner

PMI/PC



Character	Identification	Meaning	Hex
R	Read	Reading of data from the PMI/PC	52
A	Answer	Response to read telegram	41
W	Write	Writing of data in PMI/PC	57
ACK	Acknowledge	Receive data without errors	06
NAK	Negativ Acknowledge	Errors on receipt of data	15
CR	Carrige Return	End character command	0D
LF	Line Feed	End character command	0A

Character	Identification	Meaning	Hex
ETX	End of Text	End character telegram	03
ENQ	Enquiry	Start character telegram	05
ESC	Escape	Command start character	1B
SNR	Slave Number	Slave number	-
CS	Checksum	Checksum	-

With the 1:n modes, several stations (slave) can be connected to the host (master). For this reason, the station number must also be entered in the telegrams.

Note: If 255_{Dec} (FF_{Hex}) is given as a station number, the telegram is received by all control systems.



Information

A response telegram never follows a telegram with the address 255_{Dec} (FF_{Hex}) - not even ACK or NAK.

READ COMMAND IN 1:N ASCII MODE (WITH ETX/CS WITH CR + LF, WITHOUT ACK, WITHOUT NAK)

The control system with the participant address 02 should send the content of the flag words 101, 102, 103, 104 and 105 to the partner.

ENQ	SNR 2 bytes	ESC	R	Address (4 bytes)	Number of words (4 bytes)	CS 2 bytes	CR	LF
05	30 32	1B	52	30 30 36 35	30 30 30 35	45 32	0D	0A

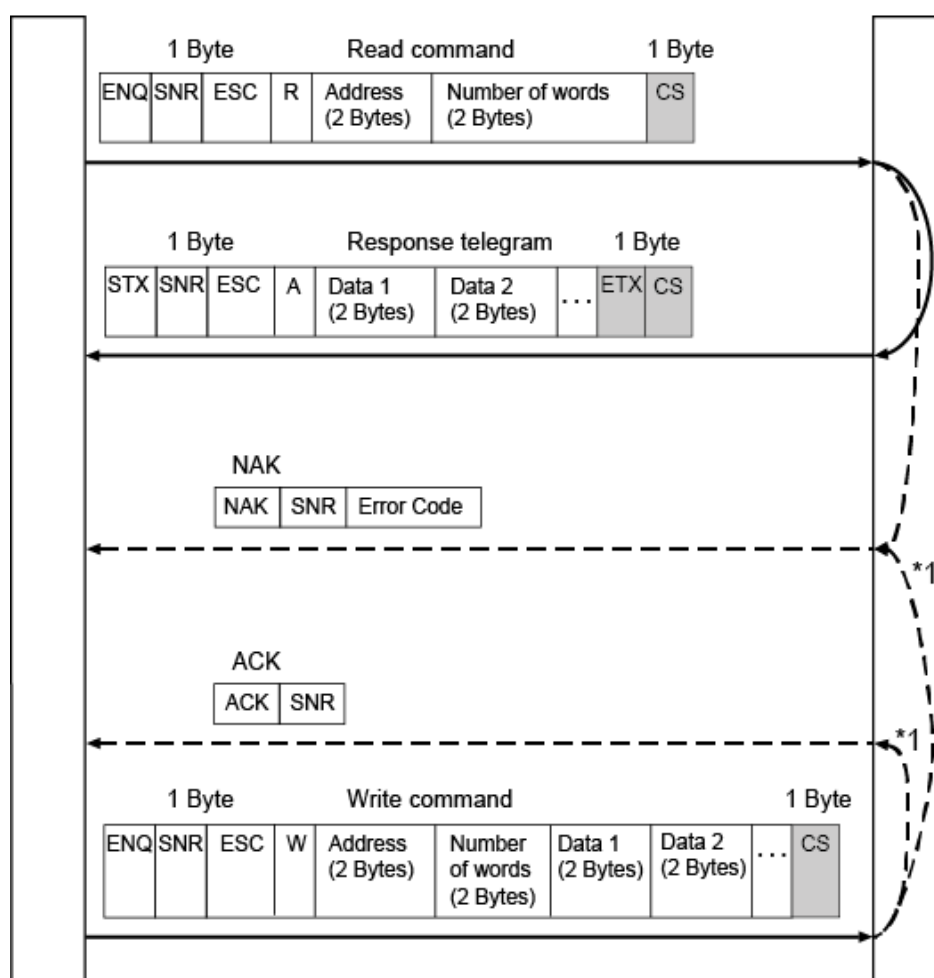
Content of	Value (dec)		Value (hex)
MW 101:	12345	3039 _{Hex}	33 30 33 39
MW 102:	6789	1A85 _{Hex}	31 41 38 35
MW103:	2	0002 _{Hex}	30 30 30 32
MW 104:	24671	605F _{Hex}	36 30 35 46
MW 105	13572	3504 _{Hex}	33 35 30 34

EN Q	SNR 2 bytes	ES C	A	MW 101	MW 102	MW 103	MW 104	MW 105	ET X	CS 2 bytes	CR	LF
05	30 32	1B	4 1	33 30 33 39	31 41 38 35	30 30 30 32	36 30 35 46	33 35 30 34	03	45 32	0D	0A

5.2.3.1.61:n Binary mode

Communications partner

PMI/PC



Character	Identification	Meaning	Hex
R	Read	Reading of data from the PMI/PC	52 H
A	Answer	Response to read telegram	41 H
W	Write	Writing of data in PMI/PC	57
ACK	Acknowledge	Receive data without errors	06 H
NAK	Negative Acknowledge	Errors on receipt of data	15 H
ETX	End of Text	End character telegram	03 H
STX	Start of Text	Start character telegram	02 H
ENQ	Enquiry	Start character telegram	05 H
ESC	Escape	Command start character	1B H
SNR	Slave Number	Slave number	-
CS	Checksum	Checksum	-

READ COMMAND IN 1:N BINARY MODE (WITH ETX/CS WITH CR + LF, WITHOUT ACK, WITHOUT NAK)

The PMI/the PC with the participant address 02 should send the content of the words 101, 102, 103, 104 and 105 to the partners.

The partner sends the following read commands:

EN Q	SNR 1 byte	ESC	R	Address (4 bytes)	Number of words (4 bytes)	CS 1 byte
05	02	1B	52	30 30 36 35	30 30 30 35	38

RESPONSE TELEGRAM IN 1:N BINARY MODE

The control system responds with the response telegram

Content of	Value (dec)	Value (hex)
MW 101:	12345	33 30 33 39
MW 102:	6789	31 41 38 35
MW103:	2	30 30 30 32
MW 104:	24671	36 30 35 46
MW 105	13572	33 35 30 34

ST X	SNR 1 byte	ES C	A	MW 101	MW 102	MW 103	MW 104	MW 105	ET X	CS 1 byte
05	02	1B	4 1	33 30 33 39	33 41 38 35	30 30 30 32	33 36 35 46	33 35 30 34	03	63

5.2.3.2 Calculation of the checksum

The checksum is for checking the telegram to see that it has been sent correctly. The data transferred can be incorrect as a result of interference. This can sometimes be detected with the checksum.

In principle, the checksum displays the sum of all transferred data.



Information

Because one byte is envisaged for the checksum, this can assume the value 255_{Dec} (FF_{Hex}) as a maximum. If the calculated sum does not exceed this value, only the last two figures of the sum are transferred in hexadecimal as a checksum.

Tip: Note the following example 2 in relation to this.

With the ASCII modes, these two figures must also be converted into ASCII characters. The Start characters *STX* and *STX* are not included in the checksum calculation. All characters after the checksum are also not included in the calculation.

5.2.3.2.1 Example for the calculation of the checksum:

EXAMPLE 1 - MODE 1:N ASCII

For mode 1:n ASCII (with ETX/CS, with CR+LF)

Writing of the value 00C8_{Hex} (200D) to address 0064_{Hex}

(100Dec) to slave 01

EN	SNR	ESC	W	MW 100	MW 200	CS	CR	LF
Q	2 bytes					2 bytes		
05	30 31	1B	57	30 30 36 34	30 30 43 38	37 38	0D	0A

Range for the calculation of the checksum

CS

$$= 30+31+1\text{B}+57+30+30+36+34+30+30+43+38$$

$$= 278_{\text{Hex}} \rightarrow 78_{\text{Hex}} \rightarrow \text{ASCII: } 37\ 38$$

EXAMPLE 2 - MODE 1:N BINARY

For mode 1:n binary (with ETX/CS, without CR+LF)

Response from the PMI/PC to read telegram, response of the values 1234_{Hex}

(4660Dec) and 4321_{Hex} (17185 dec) from slave 12_{Hex} (18Dec)

STX	SNR	ESC	A	Value 4660	Value 17185	ETX	CS
	2 bytes						1 byte
02	12	1B	41	12 34	43 21	03	1 B

Range for the calculation of the checksum

CS

= 12+1B+41+12+34+43+43+21+03 = 11BHex

= 11BHex --> only last byte: 1 BHex

6 Creating variables

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

6.1 Creating variables in the Editor

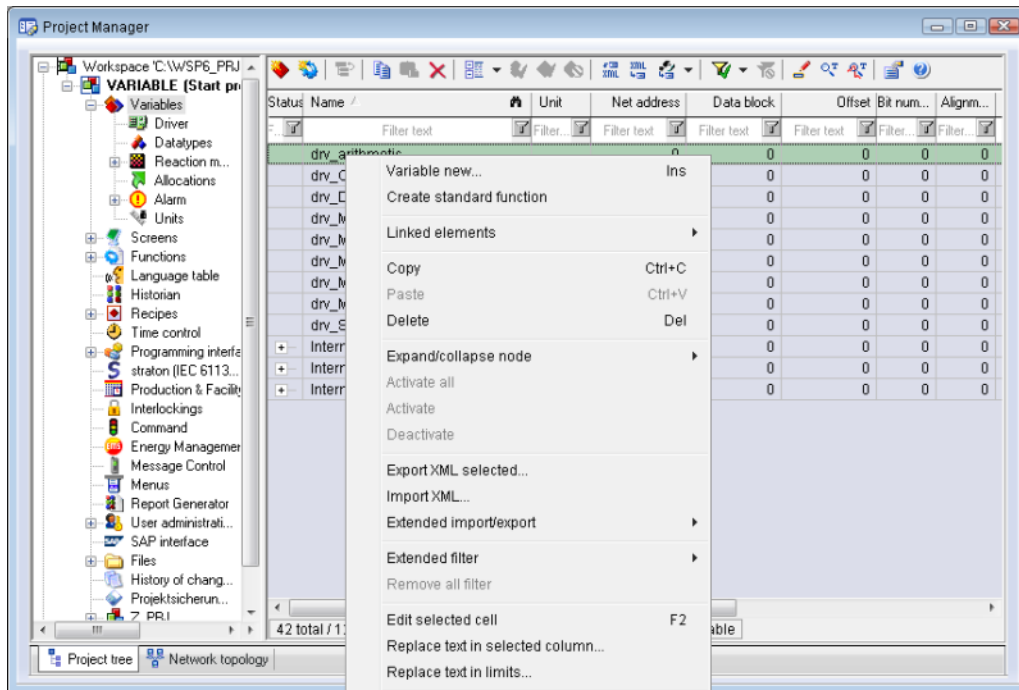
Variables can be created:

- ▶ 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:

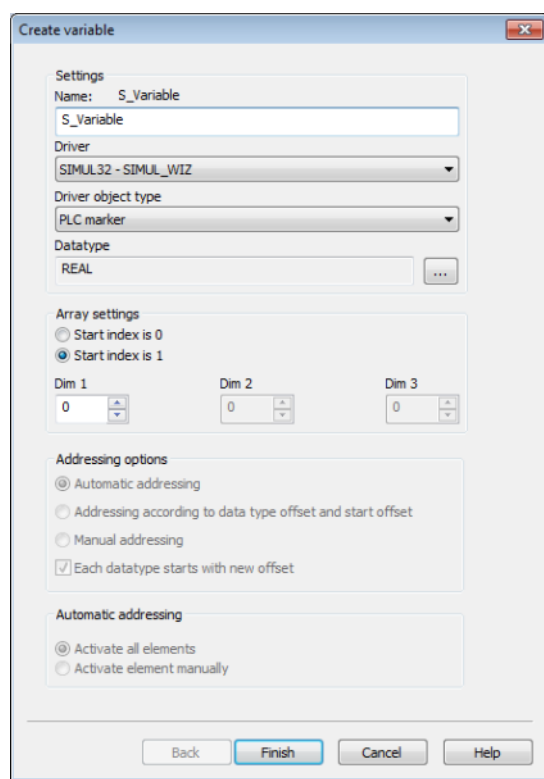
1. Select the **New variable** command in the **Variables** node in the context menu



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
- ▶ KabaDPsServer
- ▶ 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.

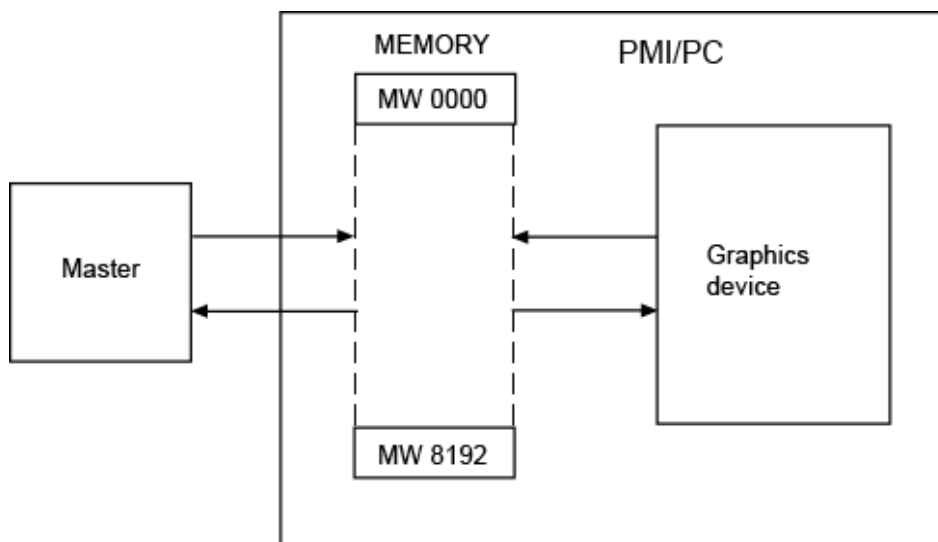
6.2 Addressing

Group/Property	Description
General	Property group for general settings.
Name	Freely definable name. Attention: For every zenon project the name must be unambiguous.
Identification	Freely definable identification. E.g. for Resources label, comments, ...
Addressing	
Net address	not used for this driver
Data block	not used for this driver
Offset	Offset of the variable; the memory address of the variable in the PLC. Possible entries: 0 to 8191. MtoM32 specific information: <ul style="list-style-type: none"> ▶ The system memory is word-orientated ▶ Each word has a low byte and a high byte. ▶ Each word has the bits 0 to 15 Attention: <ul style="list-style-type: none"> ▶ The <i>flag word 13</i> is reserved for further developments! ▶ Also note the notes on system memory (on page 37)
Alignment	not used for this driver
Bit number	Number of the bit within the configured offset. Possible entries: 0 ... 15
String length	Only available for String variables. Maximum number of characters that the variable can take. Attention: The highest offset for string flags is 8159

Group/Property	Description
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>

6.2.1 Notes on the system memory of the PMI/PC

The system memory serves as an interface for data transfer between the control system and its partner. Writing or reading to this memory is carried out by both sides (partner/visualization). The control system changes the display depending on the data in the system memory. The memory has a size of 8192 words. Addresses that go beyond this must not be used.



DESCRIPTION OF THE MEMORY ACCESS

zenon reads the data from the system memory independently. The partner determines when it needs data and requests this with a command. Upon receiving this command, zenon writes the requested data to the system memory.

- ▶ Read command:
The partner sends a read request to the control system.

- ▶ Response telegram:
zenon/the PC responds to the read command and sends the required data to the PLC.
- ▶ Write telegram:
The PLC sends data to zenon/the PC in the system memory.

6.3 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.

6.3.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channel type	Read	Write	Supported data types	Description
PLC marker	80	X	X	<i>BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING</i>	Addressing is word-orientated. UDINT is two words long and starts with each second storage word each time: 0 2 4 6 8 ...

Key:

X: supported

--: not supported

6.3.2 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.

PLC	zenon	Data type
Bit marker	BOOL	8
Byte marker	USINT	9
Byte marker	SINT	10
Word marker	UINT	2

PLC	zenon	Data type
Word marker	INT	1
DoubleWord marker	UDINT	4
DoubleWord marker	DINT	3
--	ULINT	27
--	LINT	26
Float marker	REAL	5
--	LREAL	6
String marker	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.

6.4 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.

6.4.1 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.

6.4.2 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

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

Identification	Type	Field size	Comment
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. Free string for export and display in lists.

Identification	Type	Field size	Comment
			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

Identification	Type	Field size	Comment
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.

6.5 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 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>;...;;

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

Name from import	Type	Offset	Description
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
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.

Name from import	Type	Offset	Description
MinWriteTime	UDINT	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	UDINT	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	UDINT	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	UDINT	33	Number of writing errors
ReadSucceedCount	UDINT	35	Number of successful reading attempts
MaxCycleTime	UDINT	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	UDINT	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	UDINT	26	Number of writing attempts
ReadErrorCount	UDINT	34	Number of reading errors
MaxUpdateTimeNormal	UDINT	56	Time since the last update of the priority group Normal in milliseconds (ms).
MaxUpdateTimeHigher	UDINT	57	Time since the last update of the priority group Higher in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	58	Time since the last update of the priority group High in milliseconds (ms).
MaxUpdateTimeHighest	UDINT	59	Time since the last update of the priority group Highest in milliseconds (ms).
PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

ERROR MESSAGE

Name from import	Type	Offset	Description
ErrorTimeDW	UDINT	2	Time (in seconds since 1.1.1970), when the last error occurred.

Name from import	Type	Offset	Description
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

7 Driver-specific functions

The driver supports the following functions:

8 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

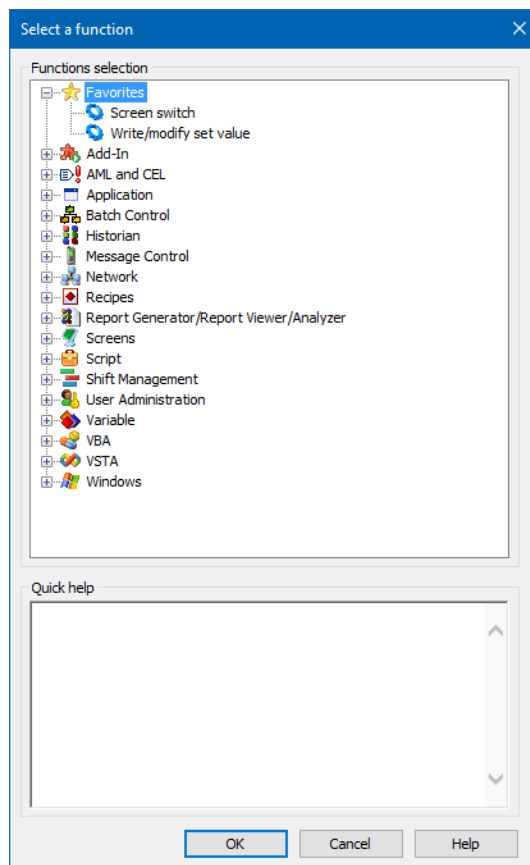
The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

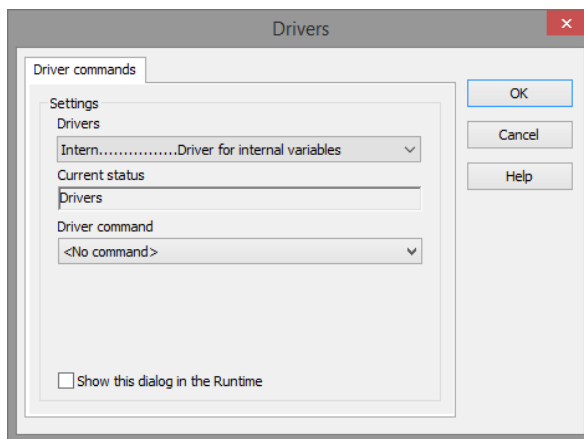
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



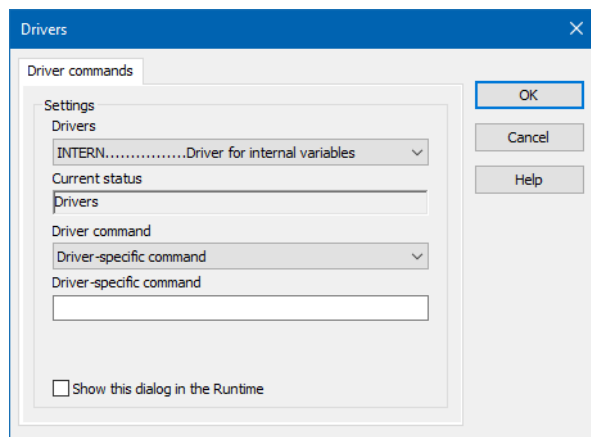
2. Navigate to the node **Variable**.
3. Select the **Driver commands** entry.

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.

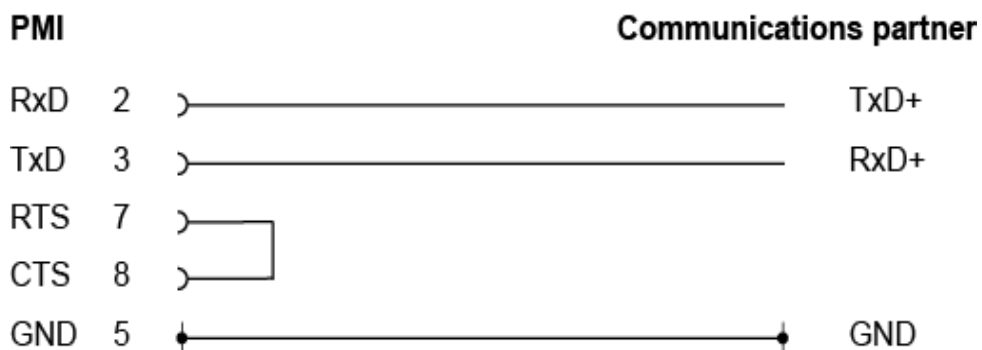
9 Connection cable

Overview of the pin assignment for the following connection cables:

- ▶ Connection cable 1 (RS 232C) (on page 57)
- ▶ Connection cable 2 (RS 422) (on page 58)
- ▶ Connection cable 3 (RS 485) (on page 58)

9.1 Connection cable 1 (RS 232C)

Pin assignment of connection cable 1 (RS 232C)

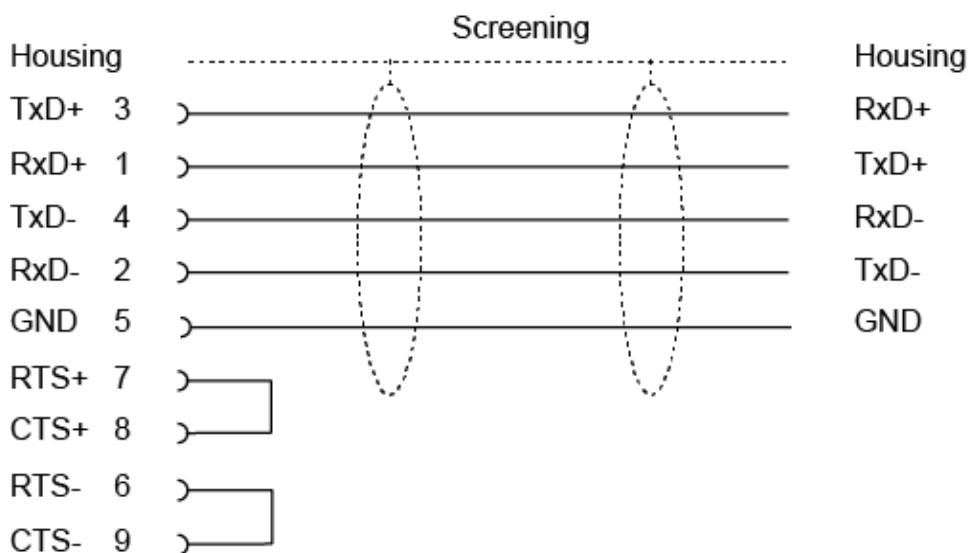


9.2 Connection cable 2 (RS 422)

Pin assignment of connection cable 2 (RS 422)

PMI

Communications partner

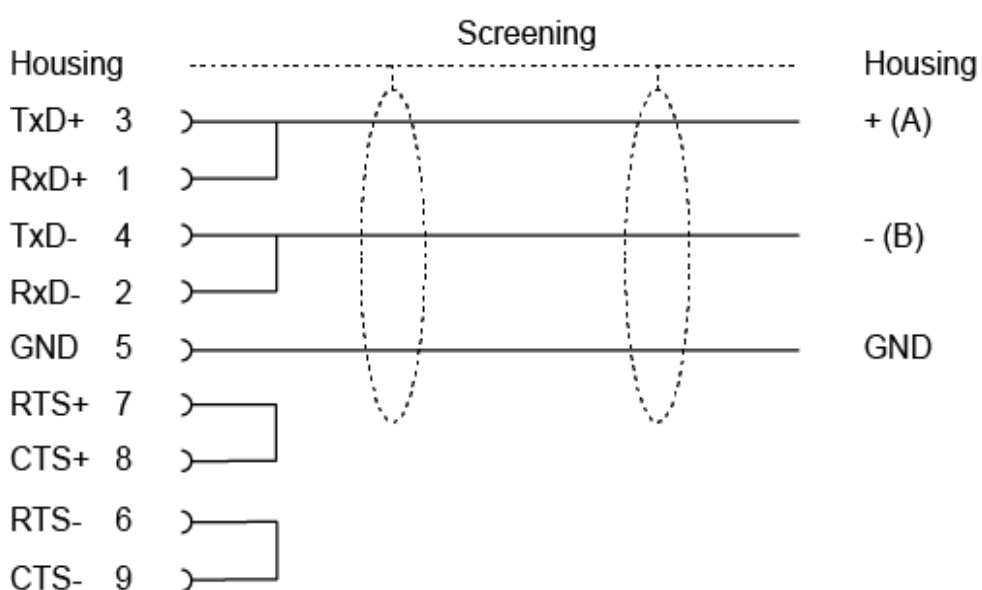


9.3 Connection cable 3 (RS 485)

Pin assignment of connection cable 2 (RS 485)

PMI

Communications partner



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 Check list

Questions and hints for fault isolation:

GENERAL TROUBLESHOOTING

- ▶ Is the PLC connected to the power supply?
- ▶ Analysis with the Diagnosis Viewer (on page 59):
-> Which messages are displayed?
- ▶ Are you using the correct cable which is recommended by the manufacturer for the connection between the PLC and the PC?
- ▶ Did you select the right COM port?
- ▶ Do the communication parameters match (Baud rate, parity, start/stop bits,...)?
- ▶ Is the COM port blocked by another application?
- ▶ Did you use the right object type for the variable?
Example: Driver variables based on driver object type **Communication details** are purely statistics variables. They do not communicate with the PLC.
You can find detailed information on this in the Communication details (Driver variables) (on page 46) chapter.
- ▶ Does the offset addressing of the variable match the one in the PLC?

VALUES ARE NOT DISPLAYED, NUMERIC VALUES REMAIN EMPTY

Driver is not working. Check the:

- ▶ Installation of zenon
- ▶ the driver installation
- ▶ The installation of all components
-> Pay attention to error messages during the start of the Runtime.

VARIABLES ARE DISPLAYED WITH A BLUE DOT

The communication in the network is faulty:

- ▶ With a network project:
Is the network project also running on the server?
- ▶ With a stand-alone project or a network project which is also running on the server:
Deactivate the property **Read from Standby Server only** in node **Driver connection/Addressing**.

VALUES ARE DISPLAYED INCORRECTLY

Check the information for the calculation in node **Value calculation** of the variable properties.

DRIVER FAILS OCCASIONALLY

Analysis with the Diagnosis Viewer (on page 59):

-> Which messages are displayed?