



COPADATA
do it your way

zenon driver manual

MATSUTCP32

v.8.00





©2018 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. MATSUTCP32 | 5 |
| 3. MATSUTCP32 - Data sheet | 6 |
| 4. Driver history | 7 |
| 5. Requirements..... | 8 |
| 5.1 PC | 8 |
| 5.2 Control | 8 |
| 6. Configuration | 9 |
| 6.1 Creating a driver..... | 9 |
| 6.2 Settings in the driver dialog | 12 |
| 6.2.1 General | 13 |
| 6.2.2 Configuration..... | 17 |
| 7. Creating variables..... | 18 |
| 7.1 Creating variables in the Editor..... | 19 |
| 7.2 Addressing..... | 22 |
| 7.3 Driver objects and datatypes | 23 |
| 7.3.1 Driver objects | 23 |
| 7.3.2 Mapping of the data types | 27 |
| 7.4 Creating variables by importing | 28 |
| 7.4.1 XML import..... | 28 |
| 7.4.2 DBF Import/Export | 29 |
| 7.5 Communication details (Driver variables)..... | 35 |
| 8. Driver-specific functions | 40 |
| 9. Driver command function | 40 |
| 10. Error analysis..... | 43 |
| 10.1 Analysis tool | 43 |

| | | |
|------|---------------------|----|
| 10.2 | Error numbers | 44 |
| 10.3 | Check list | 45 |

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. MATSUTCP32

The driver MatsuTcp32 is used for connecting to a Matsushita PLC (FP0, FP2, AFPx etc.) with an FP WEB module via Ethernet and TCP/IP.

FUNCTIONALITY

The driver gets the values of the requested variables cyclically from the configured PLCs. The communication is performed with the short (%) MEWTOCOL protocol.

3. MATSUTCP32 - Data sheet

| General: | |
|------------------|--|
| Driver file name | MATSUTCP32.exe |
| Driver name | Matsushita TCP-IP |
| PLC types | Matsushita PLCs of the types FP0/FP1/FP3/FP10SH/FP-M/FP2 |
| PLC manufacturer | Matsushita; Panasonic; |

| Driver supports: | |
|---------------------------|---------|
| Protocol | TCP/IP; |
| Addressing: Address-based | X |
| Addressing: Name-based | -- |
| Spontaneous communication | -- |
| Polling communication | X |
| Online browsing | -- |
| Offline browsing | -- |
| Real-time capable | -- |
| Blockwrite | -- |
| Modem capable | -- |
| Serial logging | -- |
| RDA numerical | X |
| RDA String | -- |

| | |
|-----------------------------------|----|
| Hysteresis | -- |
| extended API | -- |
| Supports status bit WR-SUC | -- |
| alternative IP address | -- |

| Requirements: | |
|----------------|-----------------------|
| Hardware PC | Standard network card |
| Software PC | -- |
| Hardware PLC | FP-WEB module |
| Software PLC | -- |
| Requires v-dll | -- |

| Platforms: | |
|-------------------|--|
| Operating systems | Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server 2012R2, Server 2016; |
| CE platforms | -; |

4. Driver history

| Date | Driver version | Change |
|----------|----------------|------------------------------|
| 07.07.08 | 700 | 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

| Parameters | Description |
|------------|--------------------------------------|
| Hardware | Network connection |
| Software | No additional software necessary. |
| Connection | Connection via Ethernet with TCP/IP. |

5.2 Control

| | |
|----------|---|
| Hardware | Matsushita SPS FP0, FP2, AFPx etc. with FP-WEB. |
|----------|---|

6. Configuration

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

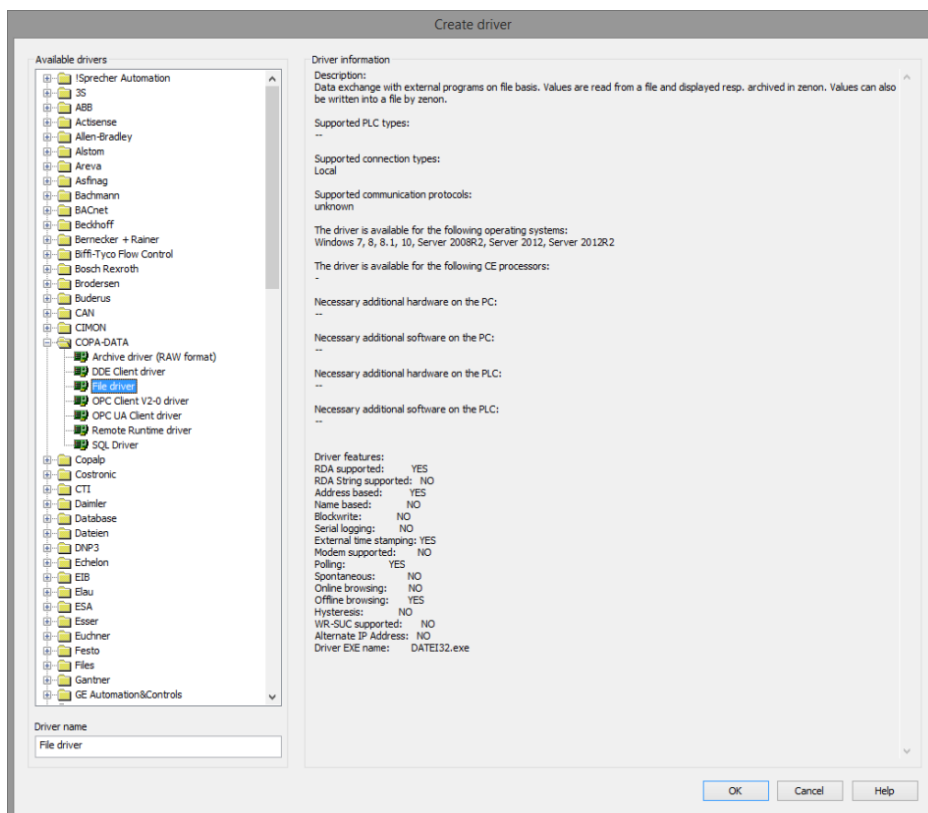


Information

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: no selection</p> |
| Driver name | <p>Unique Identification of the driver.</p> <p>Default: Empty</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: Empty</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.

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



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

The following is applicable for the **Driver name**:

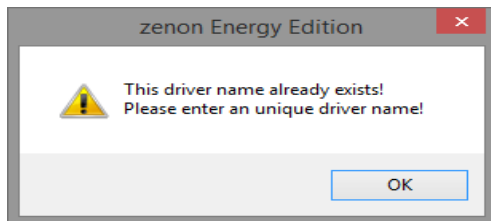
- The **Driver name** must be unique.
If a driver is used more than once in a project, a new name has to be given each time.
This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
- The **Driver name** is part of the file name.
Therefore it may only contain characters which are supported by the operating system.
Invalid characters are replaced by an underscore (_).
- **Attention:** This name cannot be changed later on.

4. Confirm the dialog by clicking on the **OK** button.
The configuration dialog for the selected driver is opened.

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

DRIVER NAME **DIALOG ALREADY EXISTS**

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



ZENON PROJECT

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

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



Information

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

6.2 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"> ▶ Hardware: A connection to the control is established. ▶ Simulation - static: No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver. ▶ Simulation - counting: No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically. ▶ Simulation - programmed: No communication is established to the PLC. The 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"> ▶ Active: Outputs can be written. ▶ Inactive: 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 mode hardware if:</p> |

- ▶ one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active

The variable image is always saved if:

- ▶ the variable is of the object type **Driver variable**
- ▶ the driver runs in simulation mode. (not programmed simulation)

The following states are not restored at the start of the Runtime:

- ▶ SELECT (8)
- ▶ WR-ACK (40)
- ▶ WR-SUC (41)

The mode **Simulation - programmed** at the driver start is not a criterion in order to restore the remanent variable image.

| | |
|-------------------------------|--|
| Stop on Standby Server | <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> <p>► Active: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status switched off (statusverarbeitung.chm::24150.htm) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.</p> <p>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> <p>► Active: The set Global update time is used for all variables in the project. The priority set at the variables is not used.</p> <p>► Inactive: The set priorities are used for the individual variables.</p> <p>Exceptions: Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the Spontaneous driver update time section.</p> |
| Priority | <p>The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.</p> <p>The variables are allocated separately in the settings of the variable properties.</p> <p>The communication of the individual variables can be graded according to importance or required topicality using the priority classes. Thus the communication load is distributed better.</p> <p>Attention: Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.</p> |

CLOSE DIALOG

| Option | Description |
|-----------|--|
| OK | Applies all changes in all tabs and closes the dialog. |

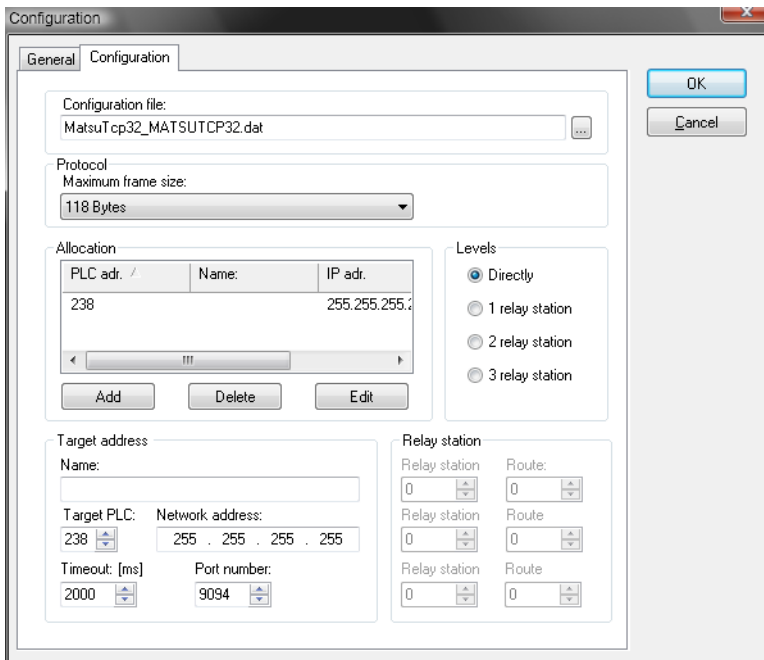
| | |
|---------------|---|
| Cancel | Discards all changes in all tabs and closes the dialog. |
| Help | Opens online help. |

UPDATE TIME FOR SPONTANEOUS DRIVERS

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

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

6.2.2 Configuration



The screenshot shows the 'Configuration' dialog box with the 'Configuration' tab selected. The 'General' tab is also visible. The 'Configuration file' field contains 'MatsuTcp32_MATSUTCP32.dat'. The 'Protocol' dropdown is set to '118 Bytes'. The 'Allocation' section shows a table with columns 'PLC adr.', 'Name', and 'IP adr.'. The table contains one entry: '238' in the 'PLC adr.' column, an empty 'Name' column, and '255.255.255.2' in the 'IP adr.' column. Below the table are 'Add', 'Delete', and 'Edit' buttons. The 'Levels' section has four radio buttons: 'Directly' (selected), '1 relay station', '2 relay station', and '3 relay station'. The 'Target address' section has a 'Name' field, a 'Target PLC' dropdown set to '238', a 'Network address' field set to '255 . 255 . 255 . 255', a 'Timeout: [ms]' field set to '2000', and a 'Port number' field set to '9094'. The 'Relay station' section has three rows, each with a 'Relay station' dropdown set to '0' and a 'Route' dropdown set to '0'. The 'OK' and 'Cancel' buttons are on the right.

| Parameters | Description |
|---------------------------|---|
| Configuration file | Enter the name of the file that will be used to store the allocation of the net address to the target PLC. If you do not enter a directory, the file will be stored and expected in the project directory. This file must exist for the runtime system. |
| Name | Freely definable name of the target PLC. Only used for informing the user. |
| Target PLC | Contains the station number of the target station. If you enter 238(0xEE), every station, independent of the configured station number, can be addressed. |
| Network address | Network address of the target station. Must be unique across the whole network! |
| Timeout | Time within which the target station must send a response to a request. |
| Port number | Number of the port that will be used for communication with the PLC via the protocol MEWTOCOL. |
| Levels | Specifies the access path via which the target station can be reached. |
| Relay station | If the target station is reached via a relais station, enter the station number of the relais station and the route number here. Enter the net address of the entry node here. |

7. Creating variables

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

7.1 Creating variables in the Editor

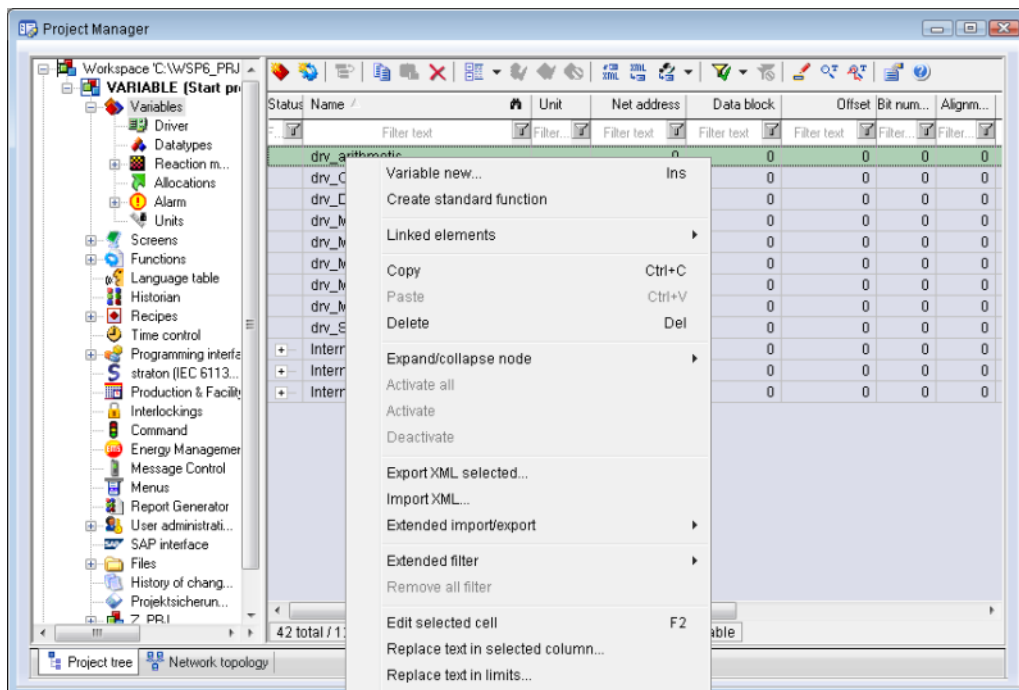
Variables can be created:

- ▶ 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:

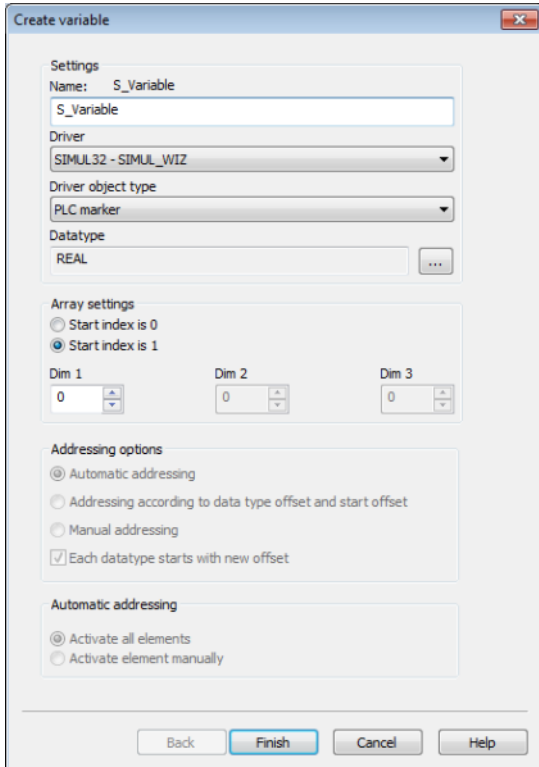
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

2. Configure the variable

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



| 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 selection dialog. |
| Array settings | Expanded settings for array variables. You can find details in the Arrays chapter. |
| Addressing options | Expanded settings for arrays and structure variables. You can find details in the respective section. |
| Automatic element activation | Expanded settings for arrays and structure variables. You can find details in the respective section. |

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.

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

| Group/Property | Description |
|---|--|
| General | Property group for general settings. |
| Name | <p>Freely definable name.</p> <p>Attention: For every zenon project the name must be unambiguous.</p> |
| Identification | <p>Freely definable identification.</p> <p>E.g. for Resources label, comments, ...</p> |
| Addressing | |
| Net address | <p>Network address of variables.</p> <p>This address relates to the target PLC of the controller, as configured for the connection configuration in the driver.</p> |
| Data block | <p>For variables of object type <code>Extended data block</code>, enter the datablock number here.</p> <p>Adjustable from 0 to 4294967295.</p> <p>You can take the exact maximum area for data blocks from the manual of the PLC.</p> |
| Offset | <p>Offset of variables. Equal to the memory address of the variable in the PLC.</p> <p>Adjustable from 0 to 4294967295.</p> |
| Alignment | not used for this driver |
| Bit number | <p>Number of the bit within the configured offset.</p> <p>Possible entries: 0 to 65535.</p> |
| String length | <p>Only available for String variables.</p> <p>Maximum number of characters that the variable can take.</p> |
| 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/Driver Object Type | <p>Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.</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> |

NET ADDRESS

The net address of the variable defines the address of the PLC via the allocation list.

7.3 Driver objects and datatypes

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

7.3.1 Driver objects

The following object types are available in this driver:

| Driver Object Type | Channel type | Read | Write | Supported data types | Comment |
|-----------------------|--------------|------|-------|---|--|
| Output | 11 | X | -- | BOOL, UDINT, DINT, USINT, UINT, INT, SINT | |
| Input | 10 | X | -- | BOOL, UDINT, DINT, USINT, UINT, INT, SINT | |
| Coupling marker | 66 | X | -- | REAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING | |
| Coupling register | 67 | X | -- | REAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING | |
| PLC data register | 20 | X | X | REAL, BOOL, DINT, UDINT, USINT, INT, UINT, SINT, STRING | |
| PLC file register | 21 | X | X | BOOL, UDINT, DINT, USINT, UINT, INT, SINT | |
| PLC marker | 8 | X | X | REAL, BOOL, DINT, UDINT, INT, UINT, USINT, SINT | |
| Timer Status | 65 | X | -- | BOOL | |
| Counter | 23 | X | X | UDINT, DINT, UINT, INT | |
| Counter status | 64 | X | -- | BOOL | |
| Timer | 22 | X | X | UDINT, DINT, UINT, INT | |
| Communication details | 35 | X | X | BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING | Variables for the static analysis of the communication; is transferred between driver and Runtime (not to the PLC). Note: The addressing and |



| | | | | | |
|--|--|--|--|--|--|
| | | | | | <p>the behavior is the same for most zenon drivers.</p> <p>You can find detailed information on this in the Communication details (Driver variables) (on page 35) chapter.</p> |
|--|--|--|--|--|--|

Key:

x: supported

--: not supported

OBJECTS FOR PROCESS VARIABLES IN ZENON

| Object | Channel type | Data type | Name |
|--------|--------------|-----------|---|
| 2 | 8 | 8 | Marker - (MX)Bit |
| 3 | 8 | 9 | Marker - (MB)Byte |
| 4 | 8 | 2 | Marker - (MW)Word |
| 5 | 20 | 2 | Data register - (MW)Word |
| 6 | 21 | 2 | File register - (MW)Word |
| 7 | 20 | 8 | Data register - (MX)Bit |
| 8 | 20 | 9 | Data register - (MB)Byte |
| 9 | 21 | 8 | File register - (MX)Bit |
| 10 | 21 | 9 | File register - (MB)Byte |
| 11 | 20 | 4 | Data register - (MD)DoubleWord |
| 12 | 21 | 4 | File register - (MD)DoubleWord |
| 13 | 8 | 4 | Marker - (MD)DoubleWord |
| 14 | 8 | 5 | Marker - (MD)REAL |
| 15 | 8 | 12 | Marker – String |
| 16 | 20 | 5 | Data register - (MD)Real |
| 17 | 20 | 12 | Data register – String |
| 18 | 22 | 2 | Timer/Counter set value - (MW)Word |
| 19 | 22 | 4 | Timer/Counter set value - (MW)DoubleWord |
| 20 | 23 | 2 | Timer/Counter actual value - (MW)Word |
| 21 | 23 | 4 | Timer/Counter actual value - (MW)DoubleWord |
| 22 | 65 | 8 | Timer Status - (MX)Bit |
| 23 | 64 | 8 | Counter Status - (MX)Bit |
| 24 | 10 | 8 | External input - (IX)Bit |
| 25 | 10 | 9 | External input - (IX)Byte |
| 26 | 10 | 2 | External input - (IW)Word |
| 27 | 10 | 4 | External input - (IX)DoubleWord |
| 28 | 11 | 8 | External output - (QX)Bit |
| 29 | 11 | 9 | External output - (QB)Byte |
| 30 | 11 | 2 | External output - (QW)Word |

| | | | |
|----|----|----|------------------------------------|
| 31 | 11 | 4 | External output - (QD)DoubleWord |
| 32 | 66 | 8 | Coupling marker - (MX)Bit |
| 33 | 66 | 9 | Coupling marker - (MB)Byte |
| 34 | 66 | 2 | Coupling marker - (MW)Word |
| 35 | 66 | 4 | Coupling marker - (MD)DoubleWord |
| 36 | 66 | 5 | Coupling marker - (MD)REAL |
| 37 | 66 | 12 | Coupling marker – String |
| 38 | 67 | 8 | Coupling register - (MX)Bit |
| 39 | 67 | 9 | Coupling register - (MB)Byte |
| 40 | 67 | 2 | Coupling register - (MW)Word |
| 41 | 67 | 4 | Coupling register - (MD)DoubleWord |
| 42 | 67 | 5 | Coupling register - (MD)REAL |
| 43 | 67 | 12 | Coupling register – STRING |

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.

| PLC | zenon | Data type |
|-----|--------|-----------|
| | BOOL | 8 |
| | USINT | 9 |
| | SINT | 10 |
| | UINT | 2 |
| | INT | 1 |
| | UDINT | 4 |
| | DINT | 3 |
| | ULINT | 27 |
| | LINT | 26 |
| | REAL | 5 |
| | LREAL | 6 |
| | STRING | 12 |

| | | |
|--|-------------------|----|
| | WSTRING | 21 |
| | DATE | 18 |
| | TIME | 17 |
| | DATE_AND_TIME | 20 |
| | TOD (Time of Day) | 19 |

Data type: The property **Data type** is the internal numerical name of the data type. It is also used for the extended DBF import/export of the variables.

7.4 Creating variables by importing

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



Information

You can find details on the import and export of variables in the Import-Export (main.chm::/13028.htm) manual in the Variables (main.chm::/13045.htm) section.

7.4.1 XML import

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.

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 cannot 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 | Char | 128 | Variable name. The length can be limited using the MAX_LAENGE entry in the project.ini file. |
| KANAL_R | C | 128 | The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered manually). The length can be limited using the MAX_LAENGE entry in the project.ini file. |
| KANAL_D | Log | 1 | The variable is deleted with the 1 entry (field/column has to be created by hand). |
| TAGNR | C | 128 | Identification. The length can be limited using the MAX_LAENGE entry in the project.ini file. |
| EINHEIT | C | 11 | Technical unit |
| DATENART | C | 3 | Data type (e.g. bit, byte, word, ...) corresponds to the 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-OBJEKTYP and DATENTYP |
| SIGMIN | Float | 16 | Non-linearized signal - minimum (signal resolution) |
| SIGMAX | F | 16 | Non-linearized signal - maximum (signal resolution) |
| ANZMIN | F | 16 | Technical value - minimum (measuring range) |
| ANZMAX | F | 16 | Technical value - maximum (measuring range) |
| ANZKOMMA | N | 1 | Number of decimal places for the display of the values (measuring range) |
| UPDATERATE | F | 19 | Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables |
| MEMTIEFE | N | 7 | Only for compatibility reasons |
| HDRATE | F | 19 | HD update rate for historical values (in sec, one decimal possible) |
| HDTIEFE | N | 7 | HD entry depth for historical values (number) |
| NACHSORT | R | 1 | HD data as postsorted values |
| DRRATE | F | 19 | Updating to the output (for zenon DDE server, in [s], one decimal possible) |
| HYST_PLUS | F | 16 | Positive hysteresis, from measuring range |
| HYST_MINUS | F | 16 | Negative hysteresis, from measuring range |
| PRIOR | N | 16 | Priority of the variable |
| REAMATRIZE | C | 32 | Allocated reaction matrix |
| ERSATZWERT | F | 16 | Substitute value, from measuring range |
| SOLLMIN | F | 16 | Minimum for set value actions, from measuring range |
| SOLLMAX | F | 16 | Maximum for set value actions, from measuring range |
| VOMSTANDBY | R | 1 | Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks |
| RESOURCE | C | 128 | Resources label. Free string for export and display in lists. The length can be limited using the MAX_LAENGE entry in project.ini . |
| ADJWVBA | R | 1 | Non-linear value adaption: 0: Non-linear value adaption is used |

| | | | |
|-----------------|---|-----|--|
| | | | 1: Non-linear value adaption is not used |
| ADJZENON | C | 128 | Linked VBA macro for reading the variable value for non-linear value adjustment. |
| ADJWVBA | C | 128 | ed VBA macro for writing the variable value for non-linear value adjustment. |
| ZWREMA | N | 16 | Linked counter REMA. |
| MAXGRAD | N | 16 | Gradient overflow for counter REMA. |



Attention

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

LIMIT 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 |
| 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** (on the installation medium in the \Predefined\Variables folder) and can be imported from there.

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 only for pure polling drivers
- ▶ Connection-related information such as ErrorMessage only for drivers that only edit one connection at a time

INFORMATION

| Name from import | Type | Offset | Description |
|-----------------------------|-------|--------|---|
| MainVersion | UINT | 0 | Main version number of the driver. |
| SubVersion | UINT | 1 | Sub version number of the driver. |
| BuildVersion | UINT | 29 | Build version number of the driver. |
| RTMajor | UINT | 49 | zenon main version number |
| RTMinor | UINT | 50 | zenon sub version number |
| RTSp | UINT | 51 | zenon Service Pack number |
| RTBuild | UINT | 52 | zenon build number |
| LineStateIdle | BOOL | 24.0 | TRUE, if the modem connection is idle |
| LineStateOffering | BOOL | 24.1 | TRUE, if a call is received |
| LineStateAccepted | BOOL | 24.2 | The call is accepted |
| LineStateDialtone | BOOL | 24.3 | Dialtone recognized |
| LineStateDialing | BOOL | 24.4 | Dialing active |
| LineStateRingBack | BOOL | 24.5 | While establishing the connection |
| LineStateBusy | BOOL | 24.6 | Target station is busy |
| LineStateSpecialInfo | BOOL | 24.7 | Special status information received |
| LineStateConnected | BOOL | 24.8 | Connection established |
| LineStateProceeding | BOOL | 24.9 | Dialing completed |
| LineStateOnHold | BOOL | 24.10 | Connection in hold |
| LineStateConferenced | BOOL | 24.11 | Connection in conference mode. |
| LineStateOnHoldPendConf | BOOL | 24.12 | Connection in hold for conference |
| LineStateOnHoldPendTransfer | BOOL | 24.13 | Connection in hold for transfer |
| LineStateDisconnected | BOOL | 24.14 | Connection terminated. |
| LineStateUnknow | BOOL | 24.15 | Connection status unknown |
| ModemStatus | UDINT | 24 | Current modem status |
| TreiberStop | BOOL | 28 | Driver stopped For <code>driver stop</code> , the variable has the value <code>TRUE</code> and an OFF bit. After the driver has started, the variable has the value <code>FALSE</code> and no OFF bit. |
| SimulRTState | UDINT | 60 | Informs the status of Runtime for driver simulation. |

| | | | |
|------------------|--------|----|--|
| ConnectionStates | STRING | 61 | <p>Internal connection status of the driver to the PLC.</p> <p>Connection statuses:</p> <p>0 : Connection OK</p> <p>1 : Connection failure</p> <p>2 : Connection simulated</p> <p>Formating:</p> <p><Netzadresse>:<Verbindungszustand>;...;;</p> <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 | BOOL | 27 | If TRUE, the modem is automatically reconnected for reading |
| ApplyCom | BOOL | 36 | Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function). |
| ApplyModem | BOOL | 37 | Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings PhoneNumberSet and ModemHwAdrSet . |
| PhoneNumberSet | STRING | 38 | Telephone number, that should be used |
| ModemHwAdrSet | DINT | 39 | Hardware address for the telephone number |

| | | | |
|----------------|--------|----|---|
| GlobalUpdate | UDINT | 3 | Update time in milliseconds (ms). |
| BGlobalUpdaten | BOOL | 4 | TRUE, if update time is global |
| TreiberSimul | BOOL | 5 | TRUE, if driver in sin simulation mode |
| TreiberProzab | BOOL | 6 | TRUE, if the variables update list should be kept in the memory |
| ModemActive | BOOL | 7 | TRUE, if the modem is active for the driver |
| Device | STRING | 8 | Name of the serial interface or name of the modem |
| ComPort | UINT | 9 | Number of the serial interface. |
| Baudrate | UDINT | 10 | Baud rate of the serial interface. |
| Parity | SINT | 11 | Parity of the serial interface |
| ByteSize | USINT | 14 | Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection. |
| StopBit | USINT | 13 | Number of stop bits of the serial interface. |
| Autoconnect | BOOL | 16 | TRUE, if the modem connection should be established automatically for reading/writing |
| PhoneNumber | STRING | 17 | Current telephone number |
| ModemHwAdr | DINT | 21 | Hardware address of current telephone number |
| RxIdleTime | UINT | 18 | Modem is disconnected, if no data transfer occurs for this time in seconds (s) |
| WriteTimeout | UDINT | 19 | Maximum write duration for a modem connection in milliseconds (ms). |
| RingCountSet | UDINT | 20 | Number of ringing tones before a call is accepted |
| ReCallIdleTime | UINT | 53 | Waiting time between calls in seconds (s). |
| ConnectTimeout | UINT | 54 | Time in seconds (s) to establish a connection. |

STATISTICS

| Name from import | Type | Offset | Description |
|----------------------|-------|--------|---|
| MaxWriteTime | UDINT | 31 | The longest time in milliseconds (ms) that is required for writing. |
| MinWriteTime | UDINT | 32 | The shortest time in milliseconds (ms) that is required for writing. |
| MaxBlkReadTime | UDINT | 40 | Longest time in milliseconds (ms) that is required to read a data block. |
| MinBlkReadTime | UDINT | 41 | Shortest time in milliseconds (ms) that is required to read a data block. |
| WriteErrorCount | UDINT | 33 | Number of writing errors |
| ReadSucceedCount | UDINT | 35 | Number of successful reading attempts |
| MaxCycleTime | UDINT | 22 | Longest time in milliseconds (ms) required to read all requested data. |
| MinCycleTime | UDINT | 23 | Shortest time in milliseconds (ms) required to read all requested data. |
| WriteCount | UDINT | 26 | Number of writing attempts |
| ReadErrorCount | UDINT | 34 | Number of reading errors |
| MaxUpdateTimeNormal | UDINT | 56 | Time since the last update of the priority group Normal in milliseconds (ms). |
| MaxUpdateTimeHigher | UDINT | 57 | Time since the last update of the priority group Higher in milliseconds (ms). |
| MaxUpdateTimeHigh | UDINT | 58 | Time since the last update of the priority group High in milliseconds (ms). |
| MaxUpdateTimeHighest | UDINT | 59 | Time since the last update of the priority group Highest in milliseconds (ms). |
| PokeFinish | BOOL | 55 | Goes to 1 for a query, if all current pokes were executed |

ERROR MESSAGE

| Name from import | Type | Offset | Description |
|------------------|------|--------|-------------|
|------------------|------|--------|-------------|

| | | | |
|-------------------|--------|----|---|
| ErrorTimeDW | UDINT | 2 | Time (in seconds since 1.1.1970), when the last error occurred. |
| ErrorTimeS | STRING | 2 | Time (in seconds since 1.1.1970), when the last error occurred. |
| RdErrPrimObj | UDINT | 42 | Number of the PrimObject, when the last reading error occurred. |
| RdErrStationsName | STRING | 43 | Name of the station, when the last reading error occurred. |
| RdErrBlockCount | UINT | 44 | Number of blocks to read when the last reading error occurred. |
| RdErrHwAdresse | DINT | 45 | Hardware address when the last reading error occurred. |
| RdErrDatablockNo | UDINT | 46 | Block number when the last reading error occurred. |
| RdErrMarkerNo | UDINT | 47 | Marker number when the last reading error occurred. |
| RdErrSize | UDINT | 48 | Block size when the last reading error occurred. |
| DrvError | USINT | 25 | Error message as number |
| DrvErrorMsg | STRING | 30 | Error message as text |
| ErrorFile | STRING | 15 | Name of error log file |

8. Driver-specific functions

The driver supports the following functions:

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

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



Information

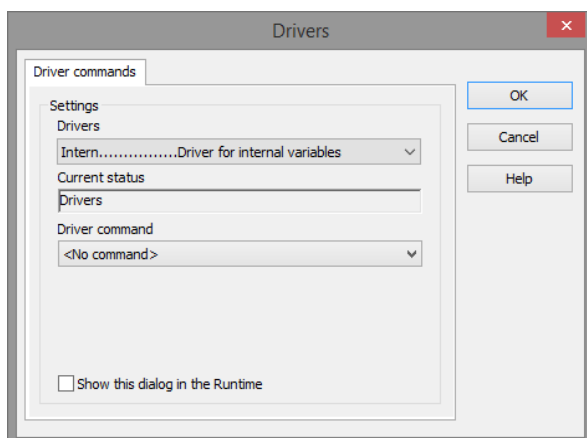
This chapter describes standard functions that are valid for most zenon drivers. However, not all functions described here are available for every driver. For example, a driver that does not, according to the data sheet, support a modem connection also does not have any modem functions.

CONFIGURATION OF THE FUNCTION

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

1. Create a new function in the zenon Editor.
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 Runtime.
Heed the notices in the **Driver command function in the network** section.

DRIVER COMMAND DIALOG



| Option | Description |
|--|--|
| Drivers | Selection of the driver from the drop-down list. It contains all drivers loaded in the project. |
| Current status | Fixed entry which has no function in the current version. |
| Driver command | Drop-down list for the selection of the command: |
| <No command> | No command is sent. A command that already exists can thus be removed from a configured function. |
| Start driver (online mode) | Driver is reinitialized and started. |
| Stop driver (offline mode) | Driver is stopped. No new data is accepted. Note: If the driver is in offline mode, all variables that were created for this driver receive the status <code>switched off (OFF; Bit 20)</code> . |
| Driver in simulation mode | Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed. |
| Driver in hardware mode | Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed. |
| Driver-specific command | Enter driver-specific commands. Opens input field in order to enter a command. |
| Activate driver write set value | Write set value to a driver is allowed. |
| Deactivate driver write set value | Write set value to a driver is prohibited. |
| Establish connection with modem | Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number. |
| Disconnect from modem | Terminate connection (for modem drivers) |
| Driver in counting simulation mode | Driver is set into counting simulation mode. All values are initialized with 0 and incremented in the set update time by 1 each time up to the maximum value and then start at 0 again. |
| Driver in static simulation mode | Driver is set into counting simulation mode. All values are initialized with 0. |
| Driver in programmed simulation mode | Driver is set into counting simulation mode. The values are calculated by a freely-programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime. |
| Show this dialog in the Runtime | The dialog is shown in Runtime so that changes can be made. |

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, which then executes the desired action on its driver. In addition, the Server sends the same driver command to the project standby. The standby also carries out the action on its driver.

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

10. Error analysis

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

10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (main.chm::/12464.htm) program that was also installed with zenon. You can find it under *Start/All programs/zenon/Tools 8.00 -> 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

DRIVER-INTERNAL ERROR CODES

| Error code | Description |
|-------------------------------|---|
| E0000000- E00000FF | The last 2 positions contain the internal Matsushita error code. Consult the documentation of the MEWTOCOL protocol for the meaning of these error codes. |
| E0000100 | Network error. More details can be found out in the attached text. |
| E0000200 | Response size does not match expected size. |
| E0000300 | The checksum in the response is incorrect. |
| E0000400 | Unexpected response character. |
| E0000500 | Error while establishing connection. |
| E0000600 | Number of received data is incorrect. |
| E0000700 | Telegram length not sufficient for checking. |

10.3 Check list

- Look at driver error file.
- Check configuration.
- Data can be read faster if it resides in only a few areas that are as close to each other as possible.