

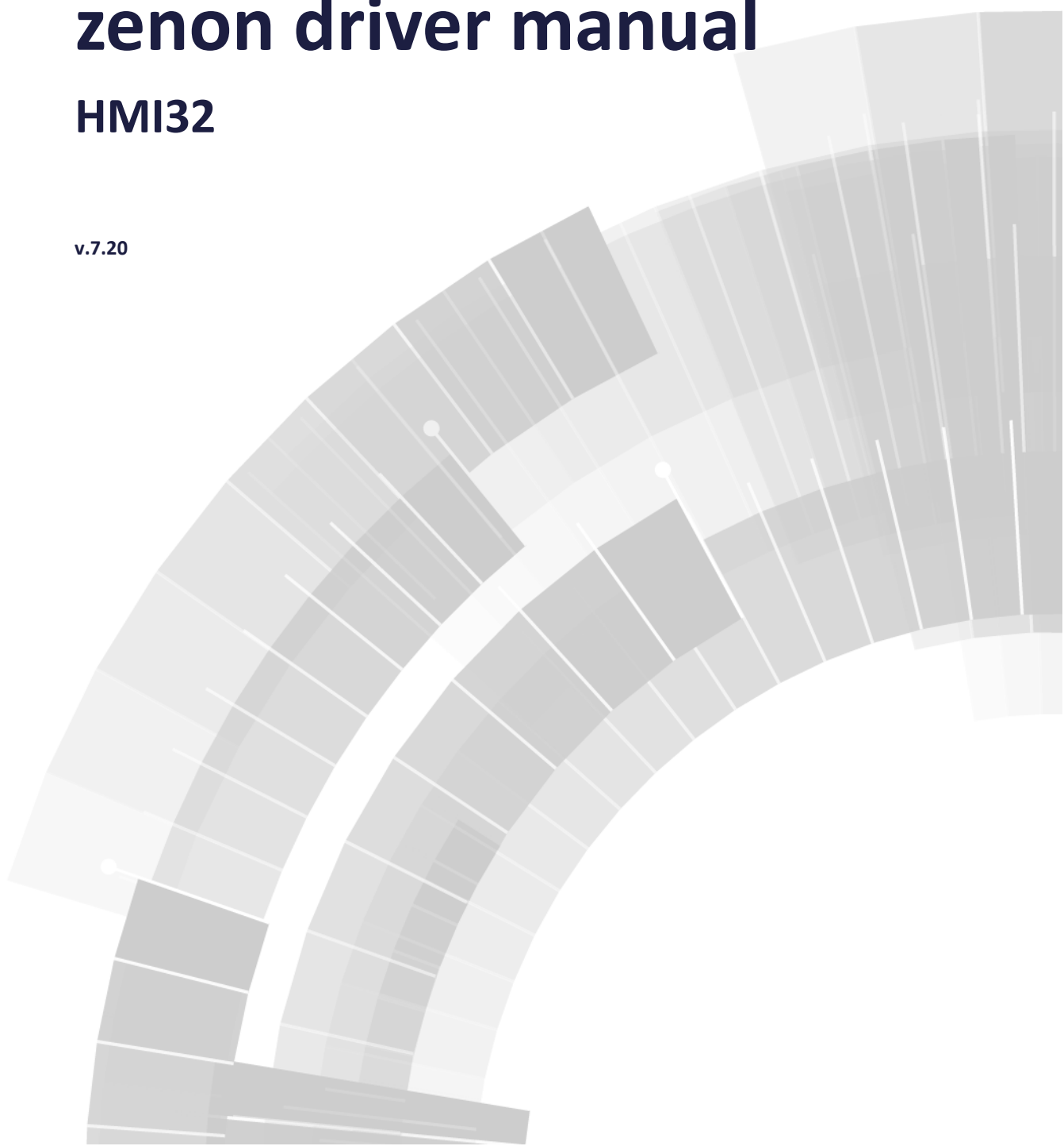


COPADATA
do it your way

zenon driver manual

HMI32

v.7.20





©2015 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

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

Contents

1. Welcome to COPA-DATA help	5
2. HMI32	5
3. HMI32 - Data sheet	6
4. Driver history	7
5. Requirements	8
5.1 PC	8
6. Configuration	8
6.1 Creating a driver	9
6.2 Settings in the driver dialog	10
6.2.1 General	11
6.2.2 Driver dialog HMI32 settings	14
7. Creating variables	15
7.1 Creating variables in the Editor	15
7.2 Addressing	19
7.3 Driver objects and datatypes	19
7.3.1 Driver objects	19
7.3.2 Mapping of the data types	22
7.4 Creating variables by importing	22
7.4.1 XML import	23
7.4.2 DBF Import/Export	23
7.5 Driver variables	29
8. Driver-specific functions	36
9. Driver commands	37
10. Error analysis	40
10.1 Analysis tool	40
10.2 Error numbers	42

10.3	Check list	45
------	------------------	----

1. Welcome to COPA-DATA help

GENERAL HELP

If you cannot find any information you require in this help chapter or can think of anything that you would like added, please send an email to documentation@copadata.com (<mailto:documentation@copadata.com>).

PROJECT SUPPORT

You can receive support for any real project you may have from our Support Team, who you can contact via email at support@copadata.com (<mailto:support@copadata.com>).

LICENSES AND MODULES

If you find that you need other modules or licenses, our staff will be happy to help you. Email sales@copadata.com (<mailto:sales@copadata.com>).

2. HMI32

The driver HMI32 is used for connecting a S7-PLC (or compatible) via the Siemens HMI adapter or Helmholz SSW7 HMI adapter.

3. HMI32 - Data sheet

General:	
Driver file name	HMI32.exe
Driver name	HMI driver
PLC types	Siemens S7-PLCs or compatible PLCs (e.g. VIPA) with S7 functions via MPI.
PLC manufacturer	Helmholz; MPI; Siemens; Vipa;

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

Requirements:	
Hardware PC	RS 232 interface; Helmholtz HMI adapter or Siemens HMI adapter; nullmodem cable for adapter PC connection.
Software PC	-
Hardware PLC	MPI compatible interface
Software PLC	-
Requires v-dll	-

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

4. Driver history

Date	Driver version	Change
07.07.08	2200	Created driver documentation

DRIVER VERSIONING

The versioning of the drivers was changed with zenon 7.10. There is a cross-version build number as of this version. This is the number in the 4th position of the file version,
For example: 7.10.0.4228 means: The driver is for version 7.10 service pack 0, and has the build number 4228.

Expansions or error rectifications will be incorporated into a build in the future and are then available from the next consecutive build number.



Example

A driver extension was implemented in build 4228. The driver that you are using is build number 8322. Because the build number of your driver is higher than the build number of the extension, the extension is included. The version number of the driver (the first three digits of the file version) do not have any significance in relation to this. The drivers are version-agnostic

5. Requirements

This chapter contains information on the requirements that are necessary for use of this driver.

5.1 PC

HARDWARE

Serial interface RS232, HMI adapter, nullmodem cable for connecting adapter and PC.

SOFTWARE

Copy the driver file HMI32.EXE into the current program directory (unless it is already there) and enter it into the file TREIBER_EN.XML with the tool driverinfo.exe.

6. Configuration

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



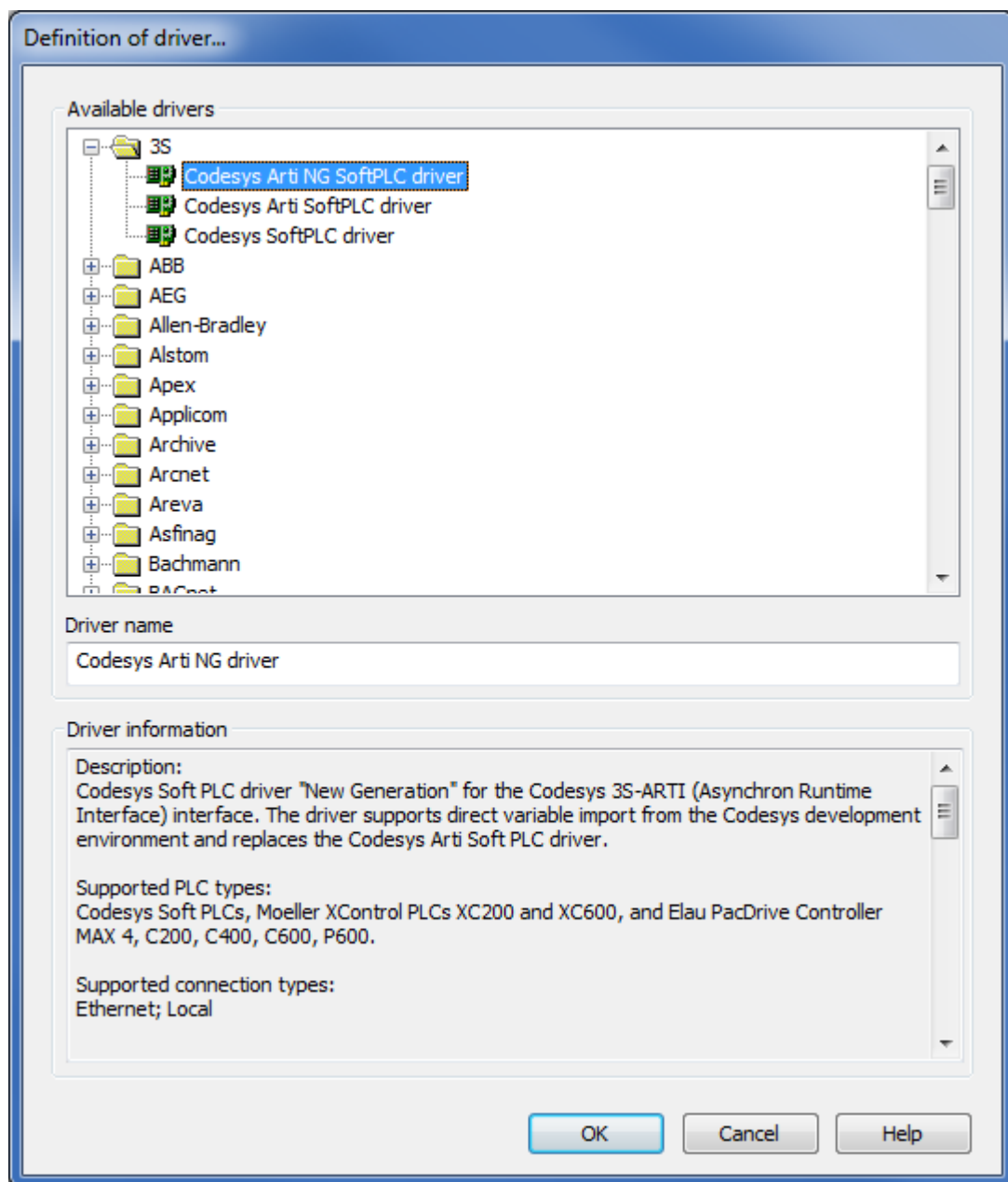
Information

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

6.1 Creating a driver

In order to create a new driver:

1. Right-click on **Driver** in the Project Manager and select **Driver new** in the context menu.
2. In the following dialog the control system offers a list of all available drivers.



3. Select the desired driver and give it a name:
 - The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.

- The driver name is part of the file name. Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (_).
 - **Attention:** This name cannot be changed later on.
4. Confirm the dialog with **OK**. In the following dialog the single configurations of the drivers are defined.

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



Information

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

- ▶ Internal
- ▶ MathDr32
- ▶ SysDrv.

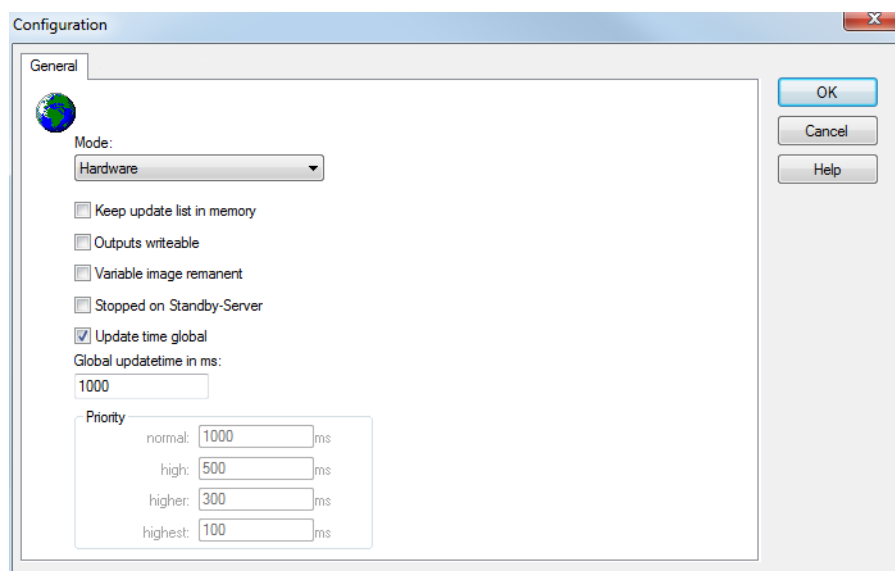
▶

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General

The configuration dialog is opened when a driver is created. In order to be able to open the dialog later for editing, double click on the driver in the list or click on the **Configuration** property.



Parameters	Description
Mode	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> ▶ Hardware: <p>A connection to the control is established.</p> ▶ Simulation static <p>No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area. E.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver.</p> ▶ Simulation - counting <p>No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically.</p> ▶ Simulation - programmed <p>N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).</p>
Keep update list in the memory	<p>Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.</p>
Output can be written	<p>Active: Outputs can be written.</p> <p>Inactive: Writing of outputs is prevented.</p> <p>Note: Not available for every driver.</p>
Variable image remanent	<p>This option saves and restores the current value, time stamp and the states of a data point.</p> <p>Fundamental requirement: The variable must have a valid value and time stamp.</p>

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

Thus the communication load is distributed better.

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

CLOSE DIALOG

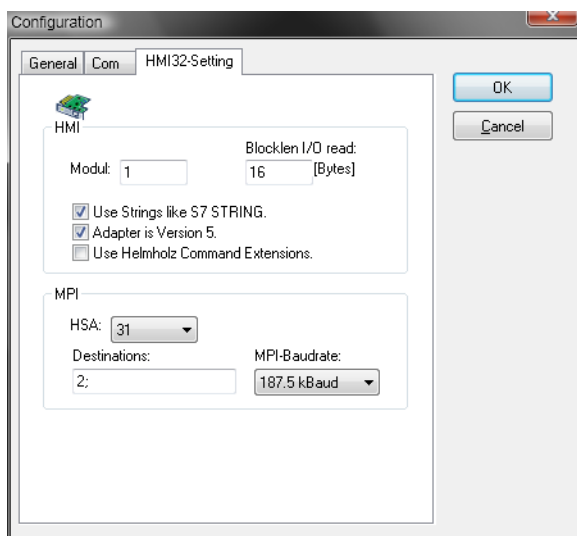
Parameters	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

UPDATE TIME FOR CYCLICAL DRIVERS

The following applies for cyclical drivers:

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

6.2.2 Driver dialog HMI32 settings



Parameters	Description
Module	MPI address of the adapter. It must be unique across the whole MPI net.
Block size	Maximum number of bytes that will be during a request at inputs and outputs.
String mapping	<p>If activated, the zenon type String will be mapped as S7 type STRING in the PLC.</p> <p>Offset + 0 is the maximum size</p> <p>Offset + 1 is the current size of the String.</p> <p>Offset + 2 first character of the text.</p> <p>Offset + n more characters of the text.</p> <p>If this option is activated, strings in zenon can have a maximum length of 61 characters, otherwise they will be truncated.</p> <p>If it is deactivated, the Strings will be handled as Char arrays.</p>
Adapter version	If the employed adapter is of version 5, this option must be activated. If it is activated, the field for entering the target address will be unlocked.
Helmholz command extension	Helmholz adapters have an extended command set, which provides for an accelerated communication to multiple MPI addresses.
HSA	Highest available MPI address in the net.
Target addresses	If the adapter is of version 5, you can enter up to 4 MPI addresses here that you want to communicate with. Use the semicolon (;) to separate the addresses.
MPI Baud rate	Transmission speed of the MPI net.

7. Creating variables

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

7.1 Creating variables in the Editor

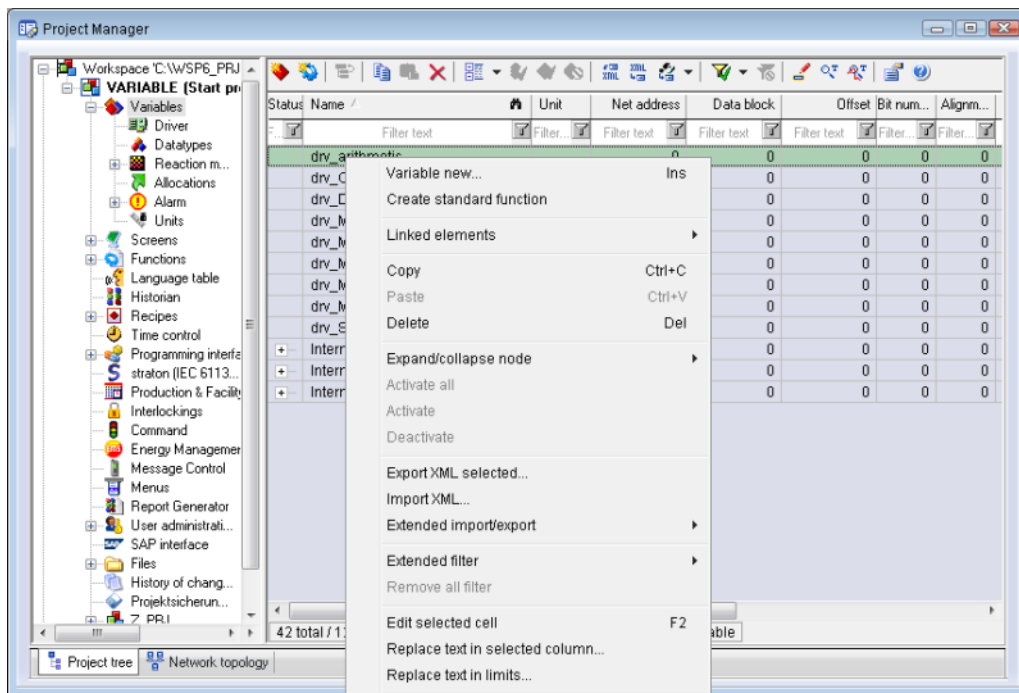
Variables can be created:

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

VARIABLE DIALOG

To create a new variable, regardless of which type:

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



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

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



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

- Settings**
 - Name: S_Variable
 - Driver: SIMUL32 - SIMUL_WIZ
 - Driver object type: PLC marker
 - Datatype: REAL
- Array settings**
 - ☐ Start index is 0
 - ☒ Start index is 1
 - Dim 1: 0
 - Dim 2: 0
 - Dim 3: 0
- Addressing options**
 - ☒ Automatic addressing
 - ☐ Addressing according to data type offset and start offset
 - ☐ Manual addressing
 - ☒ Each datatype starts with new offset
- Automatic addressing**
 - ☒ Activate all elements
 - ☐ Activate element manually

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

Property	Description
Name	<p>Distinct name of the variable. If a variable with the same name already exists in the project, no additional variable can be created with this name.</p> <p>Maximum length: 128 Zeichen</p> <p>Attention: The characters # and @ are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the Finish button remains inactive.</p> <p>Note: For some drivers, the addressing is possible over the property Symbolic address, as well.</p>
Driver	<p>Select the desired driver from the drop-down list.</p> <p>Note: If no driver has been opened in the project, the driver for internal variables (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) is automatically loaded.</p>
Driver object type (cti.chm::/28685.htm)	Select the appropriate driver object type from the drop-down list.
Data type	Select the desired data type. Click on the ... button to open the selection dialog.
Array settings	Expanded settings for array variables. You can find details in the Arrays chapter.
Addressing options	Expanded settings for arrays and structure variables. You can find details in the respective section.
Automatic element activation	Expanded settings for arrays and structure variables. You can find details in the respective section.

INHERITANCE FROM DATA TYPE

Measuring range, Signal range and Set value are always:

- ▶ derived from the datatype
- ▶ Automatically adapted if the data type is changed

Note for signal range: If a change is made to a data type that does not support the set **signal range**, the **signal range** is amended automatically. For example, for a change from **INT** to **SINT**, the **signal range** is changed to 127. The amendment is also carried out if the **signal range** was not inherited from the data type. In this case, the **measuring range** must be adapted manually.

7.2 Addressing

NET ADDRESS

The net address of the variable; defines the MPI address. If you use an adapter of version 5, you can only communicate with addresses that are entered in the target addresses field.

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	R / W	USINT, BOOL, SINT, UINT, INT	Maximum offset is different depending on the S7-CPU. See documentation on the respective Processor
Input	10	R	USINT, BOOL, SINT, UINT, INT	Maximum offset is different depending on the S7-CPU. See documentation on the respective processor
Ext. Data block	34	R / W	REAL, BOOL, WSTRING, DINT, UDINT, INT, UINT, STRING, USINT, SINT	
S5Time data block	96	R / W	REAL	Only times in seconds will be read and written. Attention: 32 bits of data will be read, but in the S7, S5Time has only 16 bits; you should therefore make sure that there are still 16 bits left after the last S5Time object in a data block.
PLC marker	8	R / W	REAL, BOOL, WSTRING, DINT, UDINT, INT, UINT, STRING, USINT, SINT	Maximum offset is different depending on the S7-CPU. See documentation on the respective Processor
Counters	23	R / W	UINT	
Time	22	R / W	REAL	
Driver variable	35	R / W	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the statistical analysis of communication. Find out more in the chapter about the Driver variables (on page 29)

OBJECTS FOR PROCESS VARIABLES IN ZENON

Object	Channel type	Data type	Name
2	8	8	Marker - (M)BOOL
3	8	9	Marker - (MB)BYTE/CHAR
4	8	2 or 1	Marker - (MW)WORD/INT

5	8	4 or 3	Marker - (MD)DWORD/DINT
6	34	8	Data block - (DBX)BOOL
7	34	9	Data block - (DBB)BYTE/CHAR
8	34	2 or 1	Data block - (DBW)WORD/INT
9	34	4 or 3	Data block - (DBD)DWORD/DINT
10	34	5	Data block - (DBD)REAL
11	34	12	Data block – STRING
12	10	8	Input - (I)BOOL
13	10	9	Input - (IB)BYTE/CHAR
14	10	2 or 1	Input - (DBW)WORD/INT
15	11	8	Output - (Q)BOOL
16	11	9	Output - (QB)BYTE/CHAR
17	11	2 or 1	Output - (QW)WORD/INT
18	22	5	Time - (T)REAL
19	8	12	Marker – STRING
21	23	2	Counter - (C)WORD
22	8	5	Marker - (MD)REAL
23	8	21	Marker - WSTRING
24	34	21	Data block - WSTRING

TIME

Depending on the employed offset, the resolution of time in the S7 can differ. The driver converts all times to seconds. You cannot write to times.

NET ADDRESS

The net address of the variable; defines the MPI address. If you use an adapter of version 5, you can only communicate with addresses that are entered in the target addresses field.

INVALID BIT

As the driver tries to send a command without errors several times after a communication error, the datapoint will only be set to invalid after 24 seconds.

7.3.2 Mapping of the data types

All variables in zenon are derived from IEC data types. The following table compares the IEC datatypes with the datatypes of the PLC.

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

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

7.4 Creating variables by importing

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



Information

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

7.4.1 XML import

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

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



Attention

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

Example:

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

7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.



Information

Import and Export via CSV or dBase supported; no driver specific variable settings, such as formulas. Use export/import via XML for this.

IMPORT DBF FILE

To start the import:

1. right-click on the variable list
2. in the drop-down list of **Extended export/import...** select the **Import dBase** command
3. follow the import assistant

The format of the file is described in the chapter File structure.



Information

Note:

- ▶ Driver object type and data type must be amended to the target driver in the DBF file in order for variables to be imported.
- ▶ dBase does not support structures or arrays (complex variables) at import.

EXPORT DBF FILE

To start the export:

1. right-click on the variable list
2. in the drop-down list of **Extended export/import...** select the **Export dBase...** command
3. follow the export assistant



Attention

DBF files:

- ▶ must correspond to the 8.3 DOS format for filenames (8 alphanumeric characters for name, 3 character suffix, no spaces)
- ▶ must not have dots (.) in the path name.
e.g. the path `C:\users\John.Smith\test.dbf` is invalid.
Valid: `C:\users\JohnSmith\test.dbf`
- ▶ must be stored close to the root directory in order to fulfill the limit for file name length including path: maximum 255 characters

The format of the file is described in the chapter File structure.



Information

dBase does not support structures or arrays (complex variables) at export.

File structure of the dBase export file

The dBaseIV file must have the following structure and contents for variable import and export:



Attention

dBase does not support structures or arrays (complex variables) at export.

DBF files must:

- ▶ conform with 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

Description	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the MAX_LAENGE entry in project.ini .
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (field/column must be entered manually). The length can be limited using the MAX_LAENGE entry in project.ini .
KANAL_D	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the MAX_LAENGE entry in project.ini .
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Bus address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADDRESS	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipe Group Manager
LES_SCHR	R	1	Write-Read-Authorization

			0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTYP and DATENTYP
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix
ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists. The length can be limited using the MAX_LAENGE entry in project.ini .
ADJWVBA	R	1	Non-linear value adaption: 0: Non-linear value adaption is used 1: Non-linear value adaption is not used

ADJZENON	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
ZWREMA	N	16	Linked counter REMA.
MAXGRAD	N	16	Gradient overflow for counter REMA.



Attention

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

LIMIT DEFINITION

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

Description	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	technical value or ID number of a linked variable for a dynamic limit (see VARIABLEx) (if VARIABLEx is 1 and here it is -1, the existing variable linkage is not overwritten)
SCHWWERT1	F	16	Threshold value for limit
HYSTERESE1	F	14	Is not used
BLINKEN1	R	1	Set blink attribute
BTB1	R	1	Logging in CEL
ALARM1	R	1	Alarm
DRUCKEN1	R	1	Printer output (for CEL or Alarm)
QUITTTIER1	R	1	Must be acknowledged
LOESCHE1	R	1	Must be deleted
VARIABLE1	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	R	1	Functions linking
ASK_FUNC1	R	1	Execution via Alarm Message List
FUNC_NR1	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
A_GRUPPE1	N	10	Alarm/event group
A_KLASSE1	N	10	Alarm/event class
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Color as Windows coding
GRENZTXT1	C	66	Limit text
A_DELAY1	N	10	Time delay
INVISIBLE1	R	1	Invisible

EXPRESSIONS IN THE COLUMN "COMMENT" REFER TO THE EXPRESSIONS USED IN THE DIALOG BOXES FOR THE DEFINITION OF VARIABLES. FOR MORE INFORMATION, SEE CHAPTER VARIABLE DEFINITION.

7.5 Driver variables

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

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

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

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



Information

Not every driver supports all driver variants.

For example:

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

INFORMATION

Name from import	Type	Offset	Description
MainVersion	UINT	0	Main version number of the driver.
SubVersion	UINT	1	Sub version number of the driver.
BuildVersion	UINT	29	Build version number of the driver.
RTMajor	UINT	49	zenon main version number
RTMinor	UINT	50	zenon sub version number
RTSp	UINT	51	zenon Service Pack number
RTBuild	UINT	52	zenon build number
LineStateIdle	BOOL	24.0	TRUE, if the modem connection is idle
LineStateOffering	BOOL	24.1	TRUE, if a call is received
LineStateAccepted	BOOL	24.2	The call is accepted
LineStateDialtone	BOOL	24.3	Dialtone recognized
LineStateDialing	BOOL	24.4	Dialing active
LineStateRingBack	BOOL	24.5	While establishing the connection
LineStateBusy	BOOL	24.6	Target station is busy

LineStateSpecialInfo	BOOL	24.7	Special status information received
LineStateConnected	BOOL	24.8	Connection established
LineStateProceeding	BOOL	24.9	Dialing completed
LineStateOnHold	BOOL	24.10	Connection in hold
LineStateConferenced	BOOL	24.11	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	BOOL	24.13	Connection in hold for transfer
LineStateDisconnected	BOOL	24.14	Connection terminated.
LineStateUnknow	BOOL	24.15	Connection status unknown
ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped For <code>driver stop</code> , the variable has the value <code>TRUE</code> and an OFF bit. After the driver has started, the variable has the value <code>FALSE</code> and no OFF bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If <code>TRUE</code> , the modem is automatically reconnected for reading
ApplyCom	BOOL	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method <code>SrvDrvVarApplyCom</code> being called (which currently has no further function).
ApplyModem	BOOL	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method <code>SrvDrvVarApplyModem</code> . This closes the current connection and opens a new one according to the settings PhoneNumberSet and ModemHwAdrSet .

PhoneNumberSet	STRING	38	Telephone number, that should be used
ModemHwAdrSet	DINT	39	Hardware address for the telephone number
GlobalUpdate	UDINT	3	Update time in milliseconds (ms).
BGlobalUpdaten	BOOL	4	TRUE, if update time is global
TreiberSimul	BOOL	5	TRUE, if driver in sin simulation mode
TreiberProzab	BOOL	6	TRUE, if the variables update list should be kept in the memory
ModemActive	BOOL	7	TRUE, if the modem is active for the driver
Device	STRING	8	Name of the serial interface or name of the modem
ComPort	UINT	9	Number of the serial interface.
Baud rate	UDINT	10	Baud rate of the serial interface.
Parity	SINT	11	Parity of the serial interface
ByteSize	USINT	14	Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection.
StopBit	USINT	13	Number of stop bits of the serial interface.
Autoconnect	BOOL	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	STRING	17	Current telephone number
ModemHwAdr	DINT	21	Hardware address of current telephone number
RxIdleTime	UINT	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)

WriteTimeout	UDINT	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	UDINT	20	Number of ringing tones before a call is accepted
ReCallIdleTime	UINT	53	Waiting time between calls in seconds (s).
ConnectTimeout	UINT	54	Time in seconds (s) to establish a connection.

STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	UDINT	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	UDINT	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	UDINT	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	UDINT	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	UDINT	33	Number of writing errors
ReadSucceedCount	UDINT	35	Number of successful reading attempts

MaxCycleTime	UDINT	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	UDINT	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	UDINT	26	Number of writing attempts
ReadErrorCount	UDINT	34	Number of reading errors
MaxUpdateTimeNormal	UDINT	56	Time since the last update of the priority group Normal in milliseconds (ms).
MaxUpdateTimeHigher	UDINT	57	Time since the last update of the priority group Higher in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	58	Time since the last update of the priority group High in milliseconds (ms).
MaxUpdateTimeHighest	UDINT	59	Time since the last update of the priority group Highest in milliseconds (ms).
PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

ERROR MESSAGE

Name from import	Type	Offset	Description
ErrorTimeDW	UDINT	2	Time (in seconds since 1.1.1970), when the last error occurred.
ErrorTimeS	STRING	2	Time (in seconds since 1.1.1970), when the last error occurred.
RdErrPrimObj	UDINT	42	Number of the PrimObject, when the last reading error occurred.
RdErrStationsName	STRING	43	Name of the station, when the last reading error occurred.
RdErrBlockCount	UINT	44	Number of blocks to read when the last reading error occurred.

RdErrHwAdresse	DINT	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	UDINT	46	Block number when the last reading error occurred.
RdErrMarkerNo	UDINT	47	Marker number when the last reading error occurred.
RdErrSize	UDINT	48	Block size when the last reading error occurred.
DrvError	USINT	25	Error message as number
DrvErrorMsg	STRING	30	Error message as text
ErrorFile	STRING	15	Name of error log file

8. Driver-specific functions

The driver supports the following functions:

The driver gets the values of the requested variables cyclically from the PLC. Depending on the employed adapter, you can communicate with a varying number of MPI addresses.

SIEMENS

Version	Description
Version 5	This adapter supports up to 4 target addresses, which must be defined in the configuration dialog.
Version 3	This adapter supports only one target address. If there are different Net addresses to communicate with, the driver will reconfigure the adapter. This reduces the transmission rate, however. Communication is possible with all addresses within the defined address area (HSA).

HELMHOLZ SSW7-HMI

This adapter supports only one target address. If there are different Net addresses to communicate with, the driver will reconfigure the adapter. The driver supports the Helmholtz-specific command

extensions, which accelerate the changing of the target address. Communication is possible with all addresses within the defined address area (HSA).

INI ENTRIES

ZENON6.INI

None

PROJECT.INI

Activating blockwrite:

[HMI32]

BLOCKWRITE=1

interface logging (also refer to interface logging)

[RS232LOG]

LOGCOM1=1

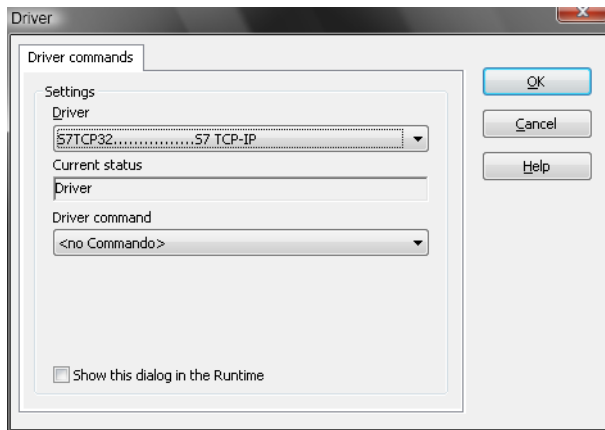
9. Driver commands

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

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

- ▶ create a new function
- ▶ select *Variables -> Driver commands*

- The dialog for configuration is opened



Parameters	Description
Drivers	Drop-down list with all drivers which are loaded in the project.
Current state	Fixed entry which has no function in the current version.
Driver commands	Drop-down list for the selection of the command.
▶ Start driver (online mode)	Driver is reinitialized and started.
▶ Stop driver (offline mode)	Driver is stopped. No new data is accepted. Note: If the driver is in offline mode, all variables that were created for this driver receive the status <code>switched off (OFF; Bit 20)</code> .
▶ Driver in simulation mode	Driver is set into simulation mode. The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▶ Driver in hardware mode	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▶ Driver-specific command	Enter driver-specific commands. Opens input field in order to enter a command.
▶ Activate driver write set value	Write set value to a driver is allowed.
▶ Deactivate driver write set value	Write set value to a driver is prohibited.
▶ Establish connection with modem	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
▶ Disconnect from modem	Terminate connection (for modem drivers)
Show this dialog in the Runtime	The dialog is shown in Runtime so that changes can be made.

DRIVER COMMANDS IN THE NETWORK

If the computer, on which the **driver command** function is executed, is part of the zenon network, additional actions are carried out. A special network command is sent from the computer to the project server, which then executes the desired action on its driver. In addition, the Server sends the same driver command to the project standby. The standby also carries out the action on its driver.

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

10. Error analysis

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

10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (main.chm::/12464.htm) program that was also installed with zenon. You can find it under *Start/All programs/zenon/Tools 7.20 -> Diagviewer*.

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

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

Attention: With the default settings, a driver only logs error information. With the **Diagnosis Viewer** you can enhance the diagnosis level for most of the drivers to "Debug" and "Deep Debug". With this the driver also logs all other important tasks and events.

In the Diagnosis Viewer you can also:

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

Note:

1. In Windows CE even errors are not logged per default due to performance reasons.
2. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.
3. The Diagnosis Viewer does not display all columns of a log file per default. To display more columns activate property **Add all columns with entry** in the context menu of the column header.
4. If you only use **Error logging**, the problem description is in column **Error text**. For other diagnosis level the description is in column **General text**.
5. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** and/or **Error code** and/or **Driver error parameter (1 and 2)**. Hints on the meaning of error codes can be found in the driver documentation and the protocol/PLC description.
6. At the end of your test set back the diagnosis level from **Debug** Or **Deep Debug**. At **Debug** and **Deep Debug** there are a great deal of data for logging which are saved to the hard drive and which can influence your system performance. They are still logged even after you close the **Diagnosis Viewer**.

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

10.2 Error numbers

Error code	Description
0000	No error
0101	Connection not established / configured
010a	Negative acknowledgement received / Timeout error
010c	Data does not exist or is locked
0200	Low resources in adapter.
0201	Wrong interface or MPI address specified
0202	Maximum number of interfaces exceeded
0204	Connections already established.
0206	Handle cannot be set
0207	Data segment cannot be locked
0209	Erroneous data field !
0302	Block too small, DW does not exist
0303	Block limit exceeded, correct number
0310	Assembly group not found
0311	Hardware error
0312	Configuration parameters incorrect
0313	Wrong Baud rate, wrong IntVector
0314	Wrong HSA
0315	MPI address error, double MPI address
0316	HW device already assigned
0317	Interrupt not available.
0318	Interrupt in use.
0319	SAP not in use.
031A	Connection error. No remote station found.
031B	Short circuit at Profibus. Syni Error.
031C	No bus profile found. System error

031D	Buffer size error.
0320	Hardware error
0330	Version conflict
0332	no response
0381	Device does not exist
0382	No driver or device found
0384	No driver or device found
03FF	System error
4001	Connection unknown
4002	Connection not established
4003	Connection is being established
4004	Connection failed
8001	Function not allowed for this operation status.
8100	Service of remote assembly group unknown.
8101	Hardware error
8103	Object access not allowed
8104	Context not supported
8105	invalid MPI address
8106	Type (data type) not supported
8107	Type (data type) not consistent
810A	Object does not exist
8301	Not enough memory on CPU
8404	serious error
8500	Wrong PDU size
8702	Address invalid
D201	Block name syntax error
D202	Function parameter syntax error
D203	Block type syntax error

D204	No embedded block available.
D205	Object already exists
D206	Object already exists
D207	Block exists in EPROM
D209	Block does not exist
D20E	No block exists
D210	Block number too high or invalid.
D241	Protection level of function not sufficient
D406	Information does not exist
EF01	Wrong ID2
FFCF	Invalid function parameters.
FFFE	unknown error FFFE hex
FFFF	Timeout error. Check interface
0800	Adapter cannot accept any new orders.

DRIVER-INTERNAL ERROR CODES

Error code	Description
1001	Too many repetitions during reception or send request not received.
1004	No data or incomplete data received.
1005	Amount of received data does not match the expected length.
1006	Too many repetitions during sending.
1007	Interface cannot be opened. Maybe specified wrong COM or COM already in use.
1008	Could not change Baud rate of the adapter communication. Maybe Baud rate invalid for this adapter model.
1009	Control system data type not supported by driver.
100A	Target address not configured. Correct the configuration.
100B	Clear to send request (STX) not received.

10.3 Check list

- ▶ Is the COM port in use by another application or are the settings incorrect?
- ▶ Is the device (PLC) that you are trying to communicate with connected to the power supply?
- ▶ Are you using a full-featured null modem cable?
- ▶ Have you checked the LEDs at the adapter and interpreted them according to the adapter documentation?
- ▶ Are the configured Net addresses available at the MPI bus? Did you specify them in the configuration?
- ▶ Are the used datablocks defined correctly in the PLC?
- ▶ Have you analyzed the driver communication error file (which errors have occurred)?

In case of communication problems, the driver writes a detailed problem analysis into the driver communication error file. This file is stored in the project directory (RT\\FILES\\zenon\\custom\\log). The name of the file is _<drivename>.txt

You can view this file with any text editor, e.g. Notepad.

For additional error analyses please send a project backup and the “error file” to the support.