zenon driver manual

IEC62056

v.7.10

Contents

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. IEC62056

The driver implements the protocol C of the IEC 62056-21 specification, but only for reading.

Only values with address information will be represented.

2.1 IEC62056 - Data sheet

General:	
Driver file name	IEC62056.exe
Driver name	IEC-62056-21 Driver
PLC types	IEC 62056-21 or IEC 61107 compatible enrgy meter
PLC manufacturer	Siemens; IEC;

Driver supports:	
Protocol	IEC 62056-21; IEC 61107;
Addressing: Address-based	-
Addressing: Name-based	x
Spontaneous communication	-
Polling communication	x
Online browsing	-
Offline browsing	-
Real-time capable	-
Blockwrite	-
Modem capable	-
Serial logging	-
RDA numerical	-
RDA String	-

Prerequisites:	
Hardware PC	Standard network card or RS232 serial interface
Software PC	-
Hardware PLC	-
Software PLC	-
Requires v-dll	-

Platforms:	
Operating systems	Windows Vista, 7, 8, Server 2008/R2, Server 2012;
CE platforms	-;

2.2 Driver history

Date	Driver version	Change
20.08.08	200	Re-created driver documentation

2.3 Requirements

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

2.3.1 PC

For communication via TCP/IP:

A standard network card in the PC is required for communication.

For serial communication:

A serial interface at the PC is required. Alternatively, you can also use a COM port server or a USB/serial converter.

2.3.2 Control

The counter can be read out via the optical interface (tested with RS232 / TVS9 converter Landis & Gyr FDC1.3)

Alternatively, you can use a serial interface module. This was not tested, however.

Ethernet communication was tested with the CU-E20 Ethernet module by Landis & Gyr.

2.4 Configuration

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

Info

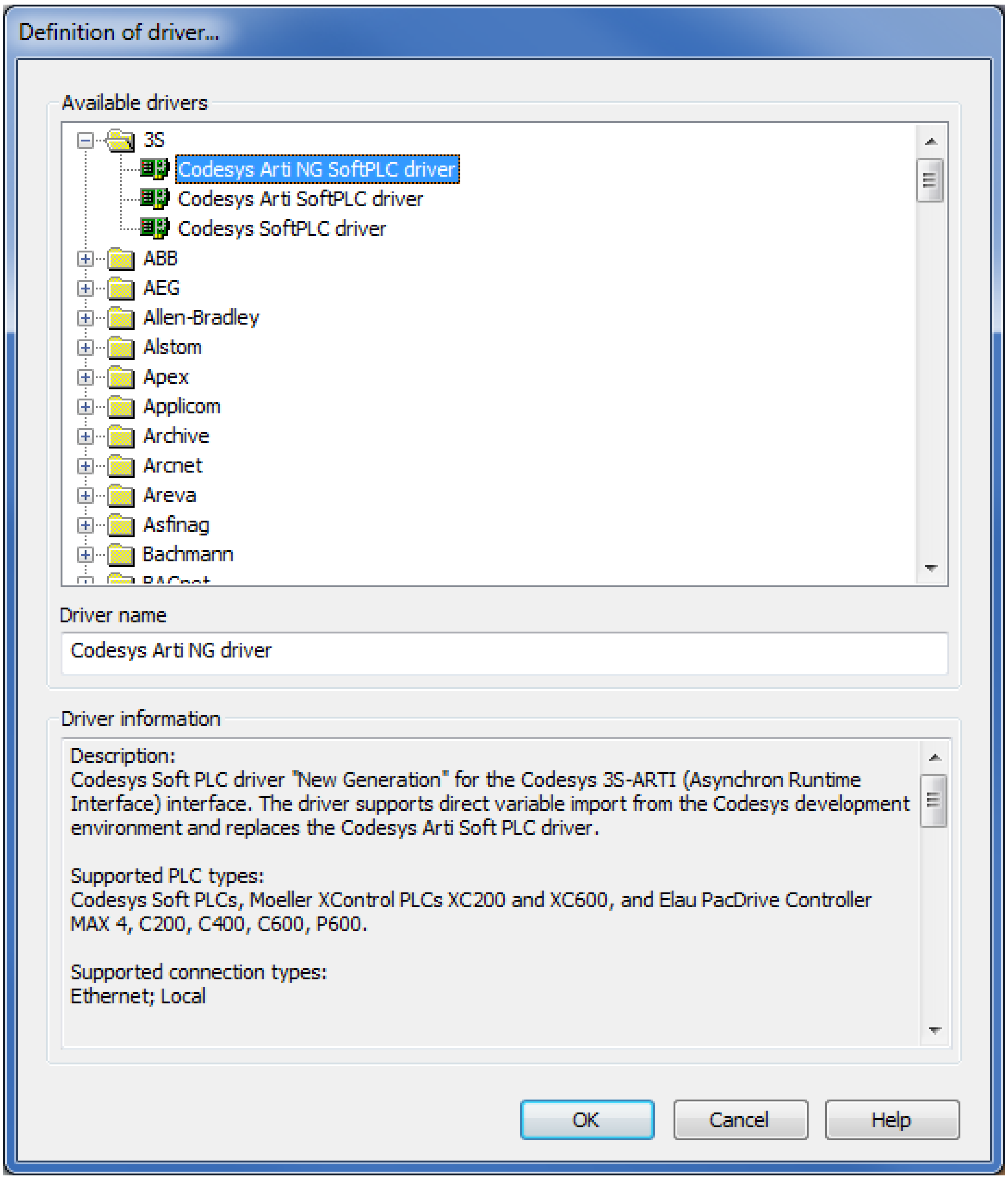
Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.

2.4.1 Creating a driver

In order to create a new driver:

1. Right-click on `Driver` in the Project Manage and select `Driver new` in the context menu.

2. In the following dialog the control system offers a list of all available drivers.



Driver name

Codesys Arti NG driver

Driver information

Description:
Codesys Soft PLC driver "New Generation" for the Codesys 3S-ARTI (Asynchron Runtime Interface) interface. The driver supports direct variable import from the Codesys development environment and replaces the Codesys Arti Soft PLC driver.

Supported PLC types:
Codesys Soft PLCs, Moeller XControl PLCs XC200 and XC600, and Elau PacDrive Controller MAX 4, C200, C400, C600, P600.

Supported connection types:
Ethernet; Local

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.

Info

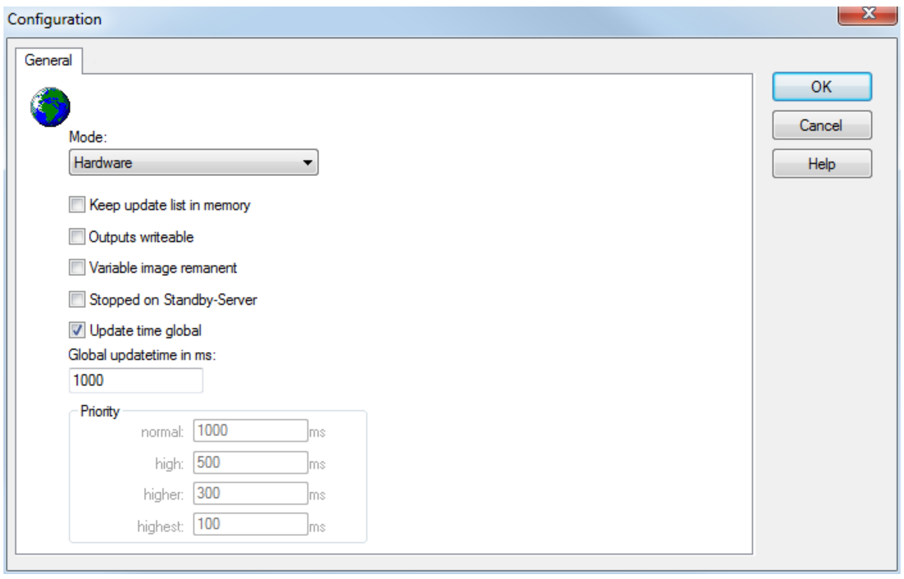
For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:

- Internal
- MathDr32
- SysDrv.

2.4.2 Settings in the driver dialog

You can change the following settings of the driver:

General



Configuration

General

Mode: Hardware

[ ] Keep update list in memory
[ ] Outputs writeable
[ ] Variable image remanent
[ ] Stopped on Standby-Server
[✓] Update time global

Global updatetime in ms: 1000

Priority
normal: 1000 ms
high: 500 ms
higher: 300 ms
highest: 100 ms

Parameter	Description
Mode	Allows to switch between hardware mode and simulation mode • 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 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).
Keep update list in the memory	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.
Output can be written	Aktiv: Outputs can be written. Inactive: Writing of outputs is prevented. **Note**: Not available for every driver.

`Variable image remanent`	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp. The variable image is saved in mode hardware if: - 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 at the Standby Server`	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. **Attention:** If this option is active, the gapless archiving is no longer guaranteed. `Aktiv`: 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.
`Global Update time`	`Aktiv`: The set `Global update time` in `ms` is used for all variables in the project. The priority set at the variables is not used. `Inactive`: The set priorities are used for the individual variables.
`Priority`	Here you set the polling times for the individual priorities. All variables with the according priority are polled in the set time. The allocation is taken

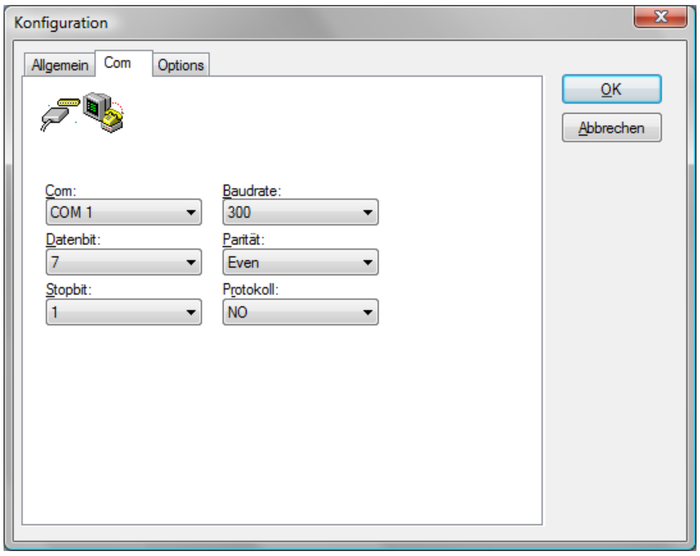
	place for each variable separately in the settings of the variable properties. The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities. Thus the communication load is distributed better.
OK	Accepts settings in all tabs and closes dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

UPDATE TIME FOR CYCLICAL DRIVER

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.

Com



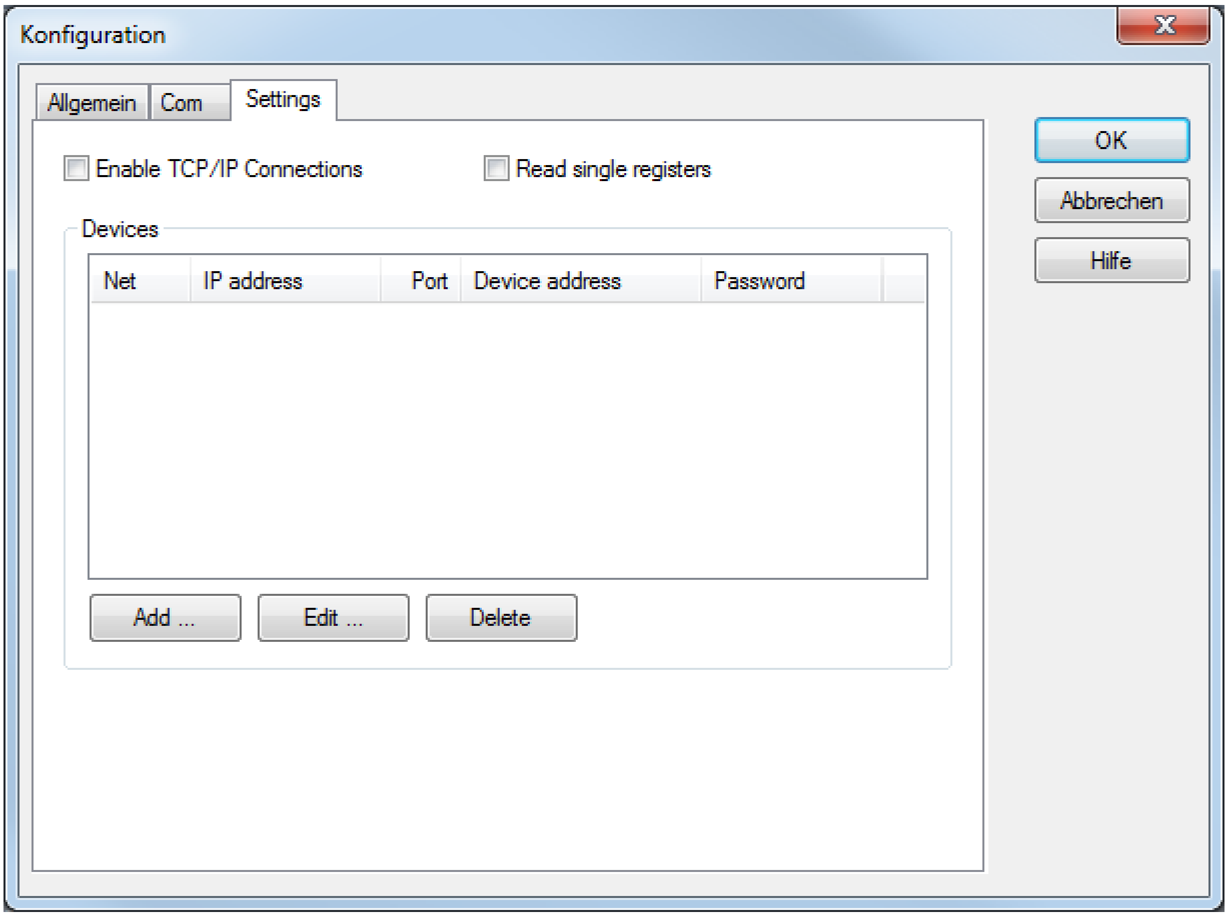
Com: COM 1
Baudrate: 300
Datenbit: 7
Parität: Even
Stopbit: 1
Protokoll: NO

According to IEC62056, the following interface parameters are to be used:

Parameters	Value
Baud rate	300
Data bit	7
Stop bit	1
Parity	Even
Protocol	NO

Settings

The driver support both serial communication and communication via TCP/IP. Only one of the two communication types can be used per device.



Enable TCP/IP Connections

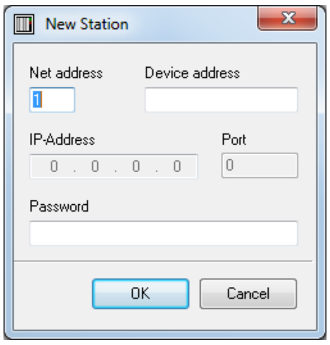
Read single registers

Devices

Net	IP address	Port	Device address	Password

Parameters	Description
Enable TCP/IP Connections	Selected: The driver communicates via TCP/IP.
Devices	List of configured stations.
Add	Opens the dialog for adding a station.
Edit	Opens the selected configuration for editing.
Delete	Deletes selected configuration.
OK	Accepts changes in all tabs and closes dialog.
Cancel	Discards changes and closes dialog.
Help	Opens online help.

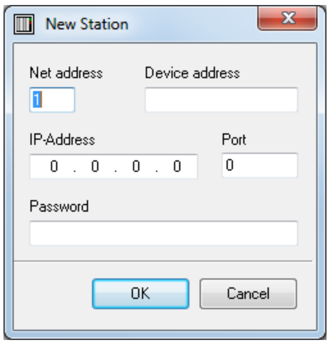
ADDING A SERIAL STATION:



Net address: 1
IP-Address: 0 . 0 . 0 . 0
Port: 0

Parameters	Description
`Net address`	You can choose any number for allocating variables to the station in the driver. The number must be unique for each driver. This number is to be given as the network address in the address settings of the variables. Area: `0-255`
`Device address`	Hardware address of the device to be read from, in accordance with IEC 62056.
`IP-Address`	Not available for serial connection
`Port`	Not available for serial connection
`Password`	Input of a password, if required.
`OK`	Accepts configuration and closes dialog.
`Cancel`	Cancels configuration and closes dialog.

ADDING A TCP/IP STATION:


New Station

Net address: 1
Device address:
IP-Address: 0 . 0 . 0 . 0
Port: 0
Password:

Parameters	Description
Net address	You can choose any number for allocating variables to the station in the driver. The number must be unique for each driver. This number is to be given as the network address in the address settings of the variables. Area: 0-255)
Device address	Not available for TCP/IP connection.
IP address	IP address of the station.
IP port	IP port of the station. Default: 1000
OK	Accepts configuration and closes dialog.
Cancel	Cancels configuration and closes dialog.

2.4.3 Communication

The driver communicates in mode Mode C, "programming mode". It can therefore address all data points in the counter. Any access password that may be necessary for this must be configured in the driver configuration for each counter.

The variable name can be configured via a driver specific property. If this property is empty, the variable identification is used as the variable name.

The time stamp can also be rounded up or down to the value of a start or end of an adjustable interval using two driver-specific properties.

To set the time via R2 command to C003, there is the `Clock Sync - BOOL` area. Writing to this variable sends the command to the counter. Variables in this area are write-only; they do not contain an actual value.

2.5 Creating variables

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

2.5.1 Creating variables in the Editor

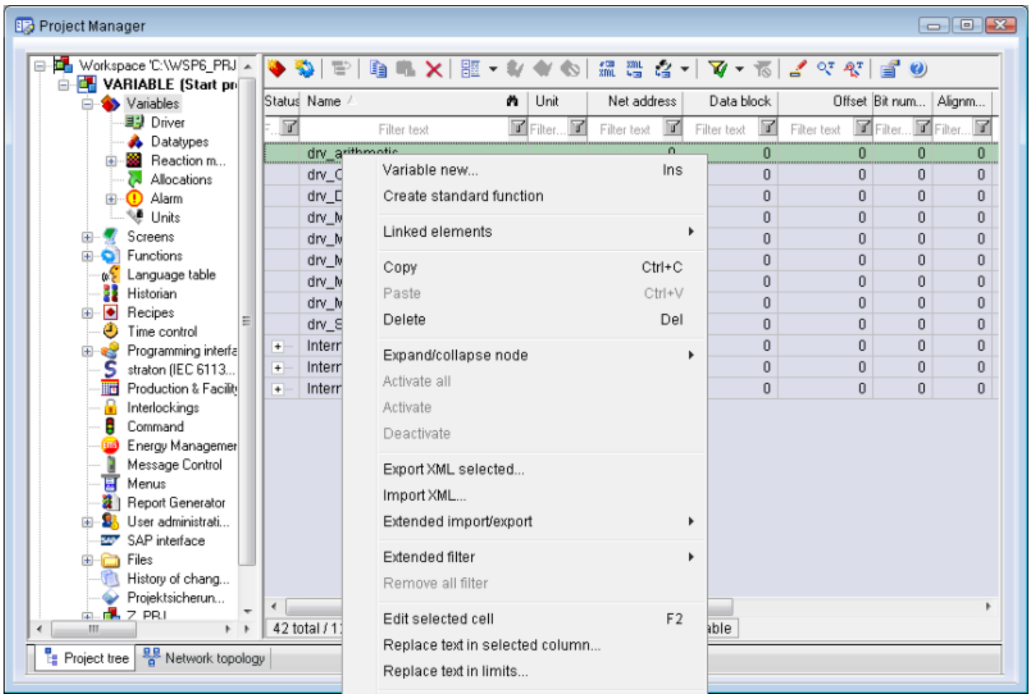
Variables can be created:

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

VARIABLE DIALOG

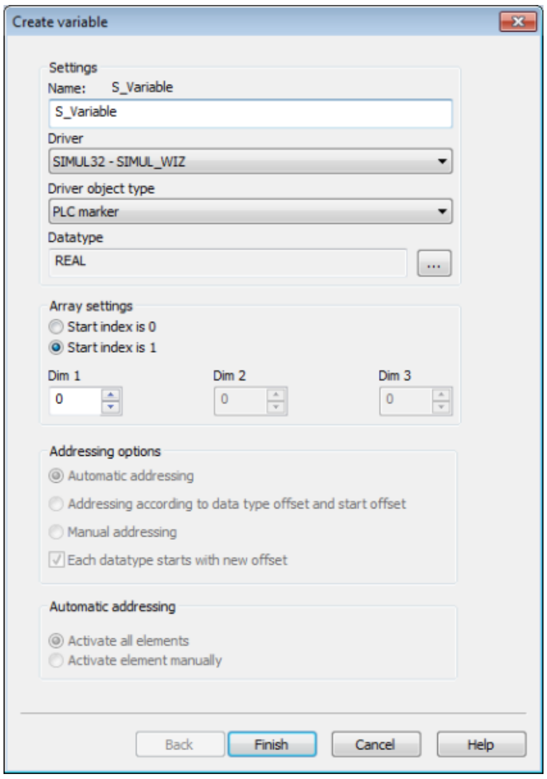
To create a new variable, regardless of which type:

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


Create variable

Settings
Name: S_Variable
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

Property	Description
Name	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. Maximum length: 128 characters **Attention:** The **#** character is not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive.
Drivers	Select the desired driver from the drop-down list. **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.
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.

2.5.2 Addressing

Property	Description
`Name`	Any name may be chosen. ATTENTION: the name must be unique within every control system project.
`Identification`	The identification field is used for addressing. (Example: 1.2.0)
`Net address`	Bus address or net address of the variable. This address refers to the bus address in the connection configuration of the driver. This defines the station, on which the variable resides.
`Data block`	not used for this driver
`Offset`	not used for this driver
`Alignment`	not used for this driver
`Bit number`	not used for this driver
`String length`	Only available for String variables: Maximum number of characters that the variable can take.
`Driver object type`	Depending on the employed driver, an object type is selected during the creation of the variable; the type can be changed here later.
`Data type`	Data type of the variable, which is selected during the creation of the variable; the type can be changed here later. ATTENTION: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.

2.5.3 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.

Driver objects

The following object types are available in this driver:

Driver object type	Channel type	Read / Write	Supported data types	Description
Input	10	R	UDINT, DINT, LREAL, STRING, DATE_AND_TIME,	
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

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	UDINT	4
DINT	DINT	3
-	ULINT	27
-	LINT	26
-	REAL	5
LREAL	LREAL	6
STRING	STRING	12
-	WSTRING	21
-	DATE	18
-	TIME	17
DATE_AND_TIME	DATE_AND_TIME	20
-	TOD (Time of Day)	19

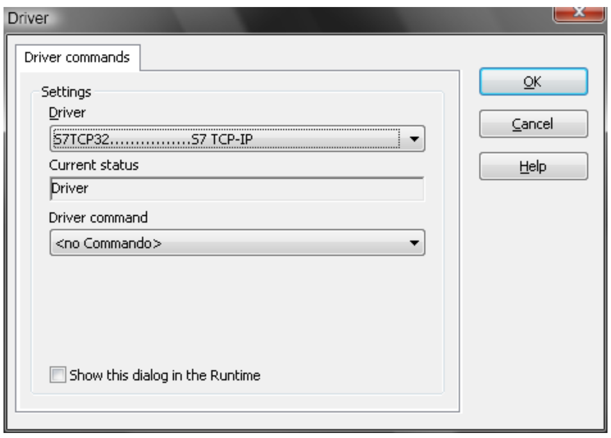
Data type: The property `Data type` is the internal numerical name of the data type. It is also used for the extended DBF import/export of the variables.

2.6 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 `switched off` (`OFF`; Bit 20).
▸ `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`	Write set value to a driver is allowed.

set value	
▸ 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.

2.7 Error analysis

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

2.7.1 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 7.10 -> 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\zenon\zenon7.10\LOG` for zenon Version 7.10. 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

Hints:

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`.

Info

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

2.7.2 Check list

Is the station connected to the power supply
Are the participants available in the TCP/IP network
Can the station be reached via the PING command
Can the station be reached via TELNET
Are the station and the PLC connected with the right cable
Did you select the right COM port
Do the communication parameters match (Baud rate, parity, start/stop bits,...)
Is the COM port blocked by another application
Did you configure the net address correctly, both in the driver dialog and in the address properties of the variable
Did you use the right object type for the variable
Does the offset addressing of the variable match the one in the station
Use the DiagViewer for further analysis -> Which messages does it show