COPADATA
do it your way

zenon driver manual

MODBUS_ENERGY

v.7.00

Contents

1. Welcome to COPA-DATA help

GENERAL HELP

If you miss any information in this help chapter or have any suggestions for additions, please feel free to contact us via e-mail: documentation@copadata.com (mailto:documentation@copadata.com).

PROJECT SUPPORT

If you have concrete questions relating to your project, please feel free to contact the support team via e-mail: support@copadata.com (mailto:support@copadata.com)

LICENSES AND MODULES

If you realize that you need additional licenses or modules, please feel free to contact the sales team via e-mail: sales@copadata.com (mailto:sales@copadata.com)

2. MODBUS_ENERGY

The Open Modbus TCP/IP driver supports Sequence of Events for energy projects and in addition (also for non energy projects) communication via a TCP/IP gateway with several slaves connected to the same serial bus.

When using a Modbus gateway, the driver can be configured so that only one TCP connection to the gateway will be established, even with several devices (slaves) behind the gateway.

The driver supports the reading of the event buffer (events) of:

- AREVA MiCOM P125/P126/P127
- COSTRONIC DFB: supports the reading of events of the Schneider Modicon TSX Premium PLC with the function block (DFB) „lynxPileHorodatageV3a" created by Costronic SA (www.costronic.ch (http://www.costronic.ch)).
- GE Multilin F650 and UR series controls
- IEC NPx800
- Schneider SEPAM

3. MODBUS_ENERGY - Data sheet

General:	
Driver file name	MODBUS_ENERGY.exe
Driver description	Modbus Energy driver
PLC types	All controllers and gateways supporting Open Modbus TCP/IP.Sequence of Events is only available for AREVA MiCOM P126/P127, GE Multilin F650, GE Multilin UR-series, ICE NPI 800, ICE NPID 800 and Schneider MODICON TSX (Premium,M340,Quantum) with Function block (DFB) from Costronic and Schneider Sepam Protection Relais.
PLC manufacturer	ABB; GE Fanuc; Modbus RTU; Mondial; Schiele; Telemecanique; Schneider; Wago; Areva; GE Multilin; ICE; Costronic;

Driver supports:	
Protocol	Modbus RTU over TCP;
Addressing: address based	x

Addressing: name based	-
Spontaneous communication	x
Polling communication	x
Online browsing	-
Offline browsing	-
Real-time capable	x
Blockwrite	x
Modem capable	-
Serial logging	-
RDA numerical	x
RDA String	x

Prerequisites:	
Hardware PC	-
Software PC	-
Hardware PLC	-
Software PLC	For Schneider MODICON TSX Premium: Function block (DFB) "cosZenonPileHorodatageV2c" from Costronic
Requires v-dll	-

Platforms:	
Operating systems	Windows CE 5.0, CE 6.0; Windows XP, Vista, 7, Server 2003, Server 2008/R2;

CE platforms	x86; ARM; Pocket-PC;

4. Driver history

Date	Driver version	Change
26.03.09	500	Created driver documentation The first release test has been conducted (GE Multilin F650 and Areva MiCOM P127)
14.10.09	900	Added COSTRONIC DFB and IEC NPx800.
07.01.10	1100	GE Multilin UR series added
09.06.10	1699	Schneider SEPAM added

5. Requirements

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

5.1 PLC

For the use of the Modbus Energy Driver, the following conditions apply:

- The PLC must support the Open Modbus TCP/IP protocol in the conformity class 0
- The reading of the event buffer is only possible with the PLCs mentioned in chapter MODBUS_ENERGY (on page 5).
- The PLCs should be set to UTC time. The driver assumes that all incoming time stamps are in UTC time.

AREVA MICOM P125/P126/P127

The time format in Areva MiCOM P12/P125/P127 Register 012Fh should be set to "Internal format" (not to "IEC format").

COSTRONIC DFB

The block should be carried out by task "FAST" and should be configured as follows.

- **ControleVie**, **IndiceEchange** and **balEvenements** must lie one behind one another, e.g. %MW900 and %MW901 and %MW902:56
- **balEvenements** must be 56 registers large
- **NombreMaxEven**... must lie between 1 and 7 (number of events in the mailbox)
- The stack for buffering of events is connected to **pile**
- **tableEvenments** contains the status of all events (1 bit per event)
- **numeroPremierEvenement** must be 0.

6. Configuration

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

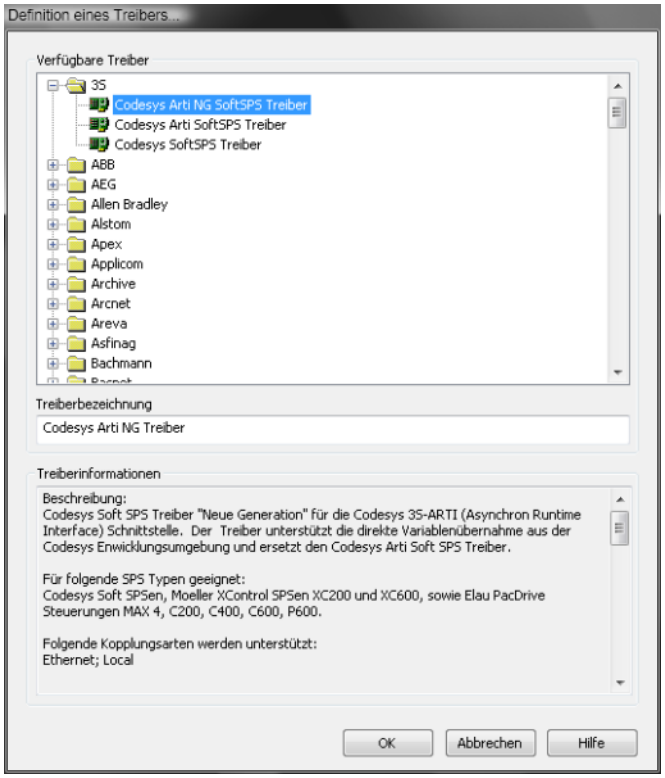
> **Info**
>
> *Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.*

6.1 Creating a driver

In order to create a new driver:

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

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



- 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, every time 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.
- 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

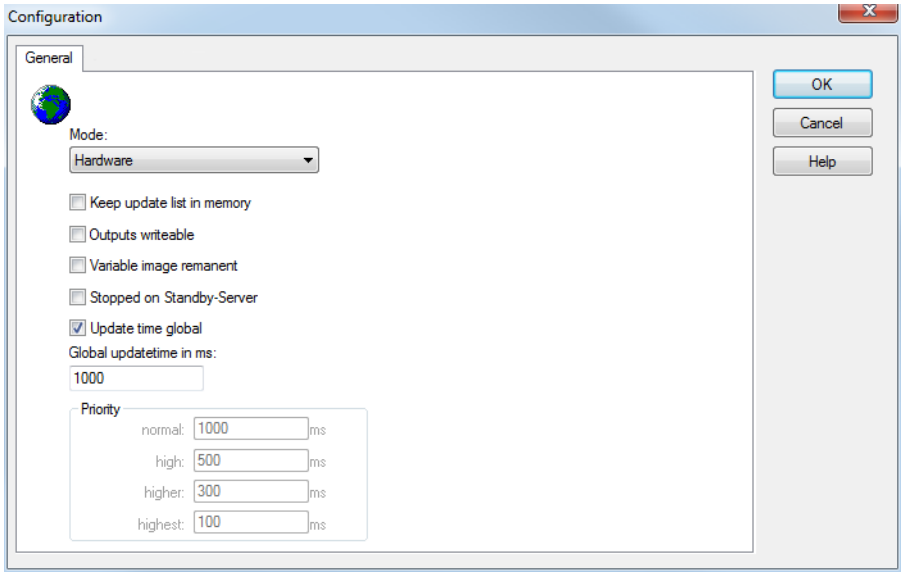
For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:

- Internal
- MathDr32
- SysDrv.

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General



Parameters	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	Active: 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. 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.
Global Update time	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. 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.

UPDATE TIME FOR CYCLICAL DRIVER

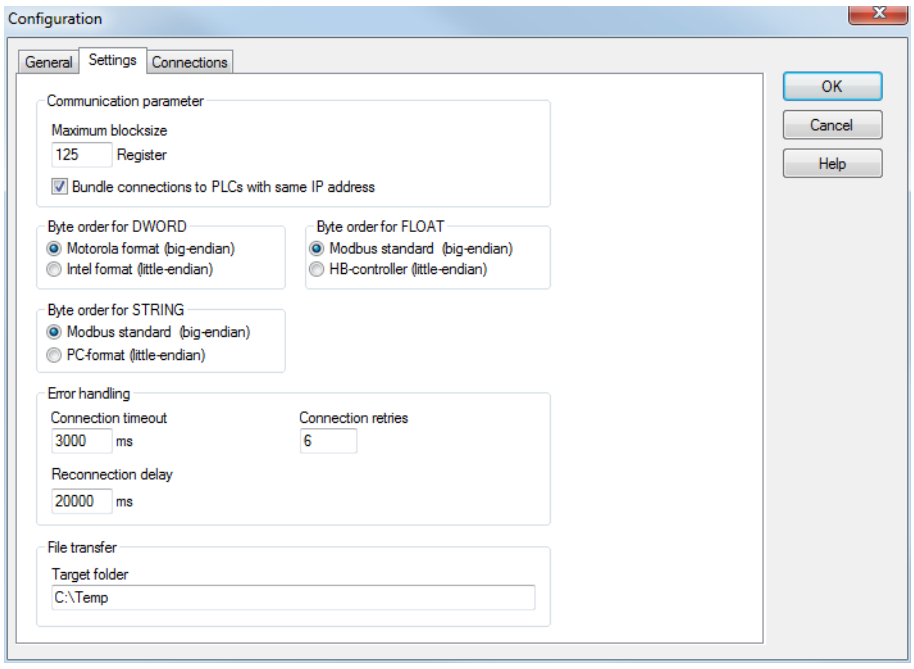
The following applies for cyclical drivers:

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.

6.2.2 Settings

You can configure the general settings for all Open Modbus TCP/IP connections in the tab **`Settings`**. To do this:

- click on the property `Configuration` in the group `General`
- now the dialog for the configuration of the driver opens.
- select the Settings tab



Parameters	Description
Communication parameter	
Maximum block size	Defines the maximum number of registers that can be queried or written using a data telegram. The configured value applies for both read and write requests. If you configure a value bigger than 100, a maximum number of 100 registers will be used for write requests. Allowed value range: 1 - 125.
Grouping connections to PLCs with the same IP address	Active: Connections with the same IP address and port number will be grouped. If you want to establish only one TCP connection to a gateway, activate this option.
Byte sequence for DWORD	Defines the sequence of lower-value and higher-value words for double word objects (DINT/UDINT). You can choose between Motorola (Big-Endian) and Intel (Little-Endian).
Motorola Format (Big-Endian	Active: DWORD ordering in accordance with Motorola format.
Intel Format (Little-Endian)	Active: DWORD ordering in accordance with Intel format.
Byte sequence FLOAT	Defines the sequence of lower-value and higher-value words for FLOAT objects (REAL). You can choose between Modbus Standard (Big-Endian) and HB-Controller (Little-Endian).
Modbus Standard (Big-Endian)	Active: Float ordering in accordance with Modbus standard.
HB Controller Float (Little-Endian)	Active: Float ordering in accordance with HB controller.
Byte sequence for STRING	Defines display of the byte sequence. **Hint:** If there is relevant information in the documentation of a PLC the following is usually applicable: MSB-first = Motorola = Big-Endian = Modbus. And: Intel = Little-Endian = PC. This can be handled differently in some documentation however.
Modbus Standard (Big-	Display in accordance with Modbus standard with switched

Endian)	characters.
PC Format (Little-Endian)	Display in accordance with Modbus standard with switched characters.
Troubleshooting	
Connection timeout	Time in milliseconds to wait for a response from the Slave. A communication error will be displayed if there is no response within this time.
Retries on error	Number of send repetitions when there is no answer from the slave after the set communication time-out. ▸ 1: a connection attempt, no repetitions ▸ 0: constant repetition ▸ Default: 6
Delay after connection termination	Time in milliseconds to wait after a communication error before trying to reestablish the connection.
Target folder	Folder for file transfer.
OK	Applies changes and closes the dialog.
Cancel	Discards changes and closes the dialog.
Help	Opens online help.

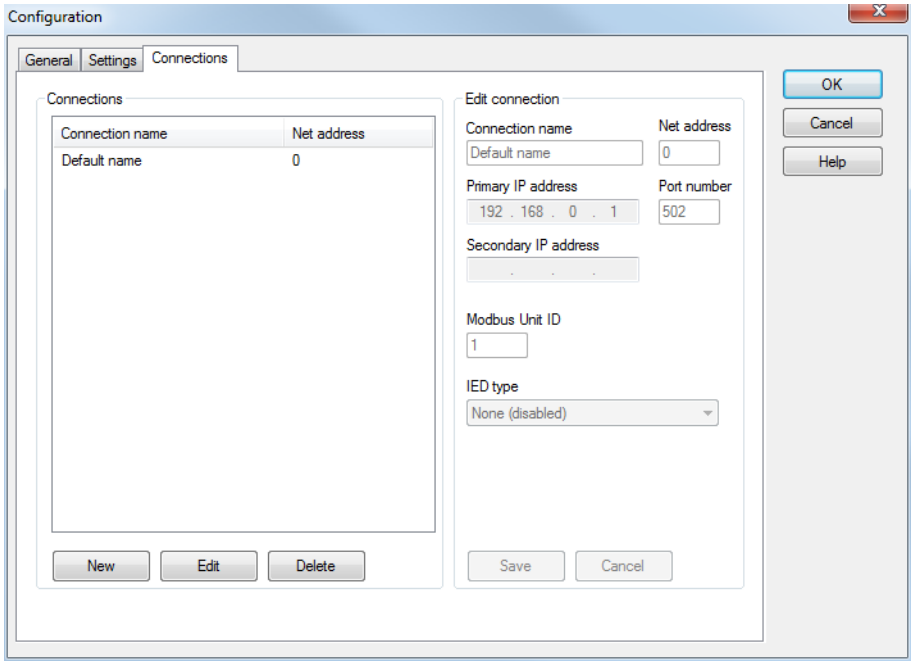
Info

Sequence of Events: If a variable is created for a connection in the event area, SOE is used. Otherwise SOE is not used.

6.2.3 Connections

You can configure the connections to the PLCs via Open Modbus TCP/IP in the tab **`Connections`**: To do this:

- click on the property `Configuration` in the group `General`
- now the dialog for the configuration of the driver opens.
- select the tab **`Connections`**



Parameters	Description
Connection list	Displays the connection names with the corresponding hardware addresses. To display the connection parameters of a name, click on the according connection name.
Net address	The net address identifies the connection. Therefore, every connection must have a unique net address. Variables are assigned to a connection via the net address. At the creation of a new variable the net address with the highest existing address + 1 is initialized.
Connection name	Freely definable name for the easier distinction of connections. **Attention:** The connection name may not contain any of the following characters: { } | & ~ ! [ ] ( ) " ' ; =
Primary IP address	Primary IP address of the PLC with which you are communicating.
Port number	The port number of the PLC. Default for Open Modbus TCP/IP: 502
Secondary IP address	Secondary IP address of the PLC with which you are communicating. If a connection to the first address cannot be achieved, a connection to the secondary IP address is tried.
Modbus Unit ID	Modbus Unit ID (Slave address)
IED type	Selection of the IED type (IED = Intelligent Electronic Device). A left-click opens the drop-down list for selection: ▸ AREVA MiCOM P126/127 ▸ COSTRONIC DFB ▸ GE Multilin F650 ▸ GE Multilin UR series ▸ ICE NPx800 ▸ Schneider SEPAM ▸ None (disabled) - Standard Modbus TCP/IP Default: None (disabled)
New	Creates a new connection with default settings.

Delete	Deletes the connection selected in the connection list.
Edit	Opens the selected connection for editing.
Save	Saves a new or edited connection.
Cancel	Cancels the editing of connection settings without saving changes.

ENHANCEMENT WHEN SELECTING COSTRONIC DFB

Parameters	Description
Base register Mailbox	Address of ControleVie.
Base register Events	Base address of tableEvenments.
Number of registers	Length of the array which is connected to tableEvenments (number of the Event bit/16).

CREATE NEW CONNECTION

1. click on the button **New**
2. enter the connection parameters Net address, Connection name and Modbus Unit ID
3. Choose the IED type if necessary
4. Click on **Save**

Note: If a established connection is selected at the creation, its configuration is copies.

DISPLAY CONNECTION PARAMETERS OF A CONNECTION

1. select the desired connection in the connection list with the mouse pointer.
2. the parameters will be displayed

EDIT CONNECTION

1. select the connection in the connection list

2. click on the button **Edit**
3. change the connection parameters
4. finish with **Save**

DELETE CONNECTION

1. select the connection in the connection list
2. click on the button **Delete**
3. the connection will be removed from the list

7. Creating variables

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

7.1 Creating variables in the Editor

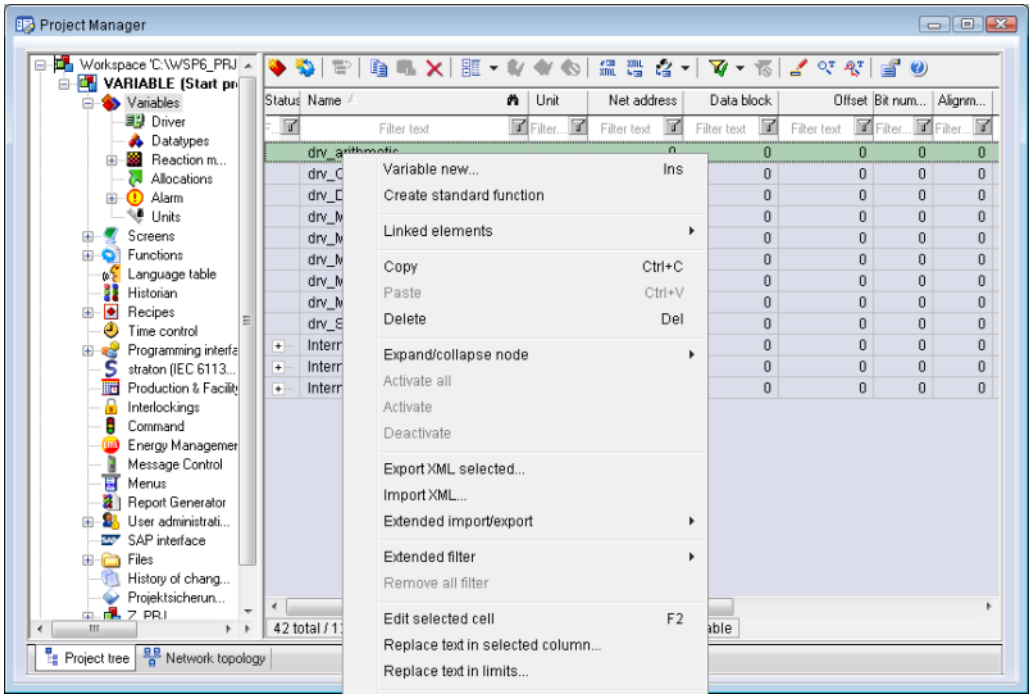
Variables can be created:

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

VARIABLE DIALOG

To create a new variable, regardless of which type:

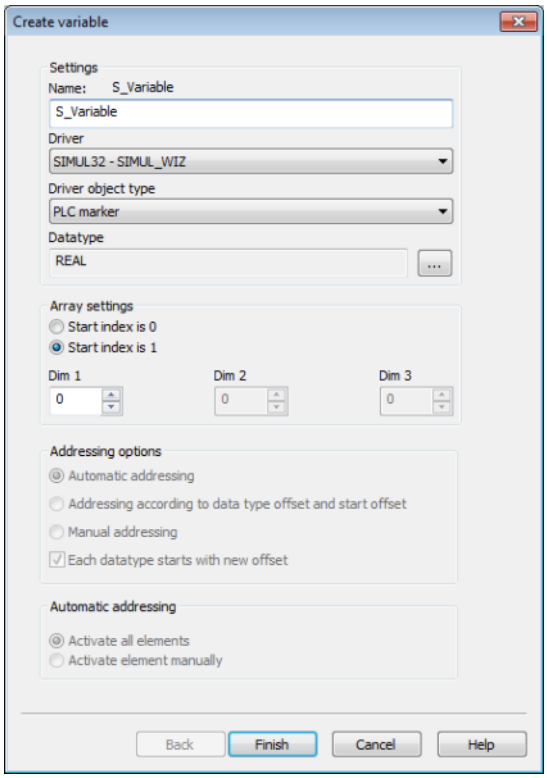
1. Select the **`New variable`** command in the **`Variables`** node in the context menu



2. The dialog for configuring variables is opened

3. configure the variable

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



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. **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.

7.2 Addressing

You define the addressing of the variables in the property window:

Group/Property	Description
General	
Name	Freely definable name. **Attention:** For every zenon project the name must be unambiguous.
Identification	Any text can be entered here, e.g. for resource labels, comments ...
Addressing	
Net address	This address refers to the bus address in the connection configuration of the driver. Specifies, on which PLC the variable resides.
Data block	not used for this driver

Offset	The meaning is dependent on the setting of the driver object types. ▸ For variables of the driver object type **Register** or **SOE - Register event**: Reference number of the register [0 to 65535] ▸ For variables of the driver object type **SOE - Numbered event**: Identifies the event. Range of values for AREVA MiCOM P125/P126/P127: 1 to 127 or 162 Range of values for Constronic DFB: 0 to number of registers Value area for GE Multilin F650: 0 to 191 Range of values for GE Multilin UR: 0 to number of FlexOperandStates Value area for Schneider SEPAM: 1000h to 105Fh
Alignment	Alignment for variables with byte length 1. You can choose between low byte and high byte.
Bit number	Number of the bit within the configured offset. Possible entries: 0 ... 15
String length	Only available for String variables: Maximum number of characters that the variable can take.
Driver connection/Driver object type	Allows you to change the driver object type which was selected during the creation of the variable.
Driver connection/Data type	Allows you to change the data type which was selected during the creation of the variable. ATTENTION: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.

7.3 Driver objects and datatypes

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

7.3.1 Driver objects

The following object types are available in this driver:

Driver-object type	Channel type	Read: Modbus function (hex/dec code)	Write: Modbus function (hex/dec code)	Supported data types	Comment
Register	8	0x03 / 3	0x10 / 16	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, STRING	Class 0 - multiple registers. Linear addressing: ▸ Bit: one-step via offset and bit number ▸ Byte (8 bits): one-step via offset and orientation ▸ Word (16 bits) Double word (32 bits) Float (32 bits) String(n*Byte): one-step via offset
Coil	65	0x01 / 1	0x05 / 5	BOOL	Class 1 - coils. Linear addressing one-step via offset.
Input discrete	66	0x02/2	N/A	BOOL	Class 1 - input discretes. Linear addressing one-step via offset.
Input register	64	0x04 / 4	N/A	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, STRING	Class 1 - input registers. Linear addressing as with registers.
SOE – Numbered event	10	Y	N/A	BOOL	PLC specific. Sequence of Events (read event buffer): Addressing of events takes place according to event

					number.
SOE – Register event	11	Y	N/A	BOOL, USINT, UINT	PLC specific. Sequence of Events (read event buffer): Addressing according to the address of the corresponding Modbus register allocated to the event.
File transfer	12	Y	Y	STRING	PLC specific.
Driver variable	35	Y	Y	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the statistical analysis of communication. **Note:** Transfers between Runtime and driver not to the PLC. Find out more in the chapter about the Driver variables (on page 36)

MODBUS FUNCTION CODES

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

0x06 / 6	Write single register	This function code is used to write a single (one word) holding register in a remote device.
0x10 / 16	Write multiple registers	This function code is used to write a block of contiguous holding registers (1 to approx. 120 words) in a remote device.
0x11 / 17	Report Slave ID	This function code is used to read the description of the type, the current status, and other information specific to a remote device.
0x2B / 43	Read Device Identification	This function code allows reading the identification and additional information relative to the physical and functional description of a remote device.

7.3.2 Mapping of the data types

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

PLC	zenon	Data type
Register / Event	BOOL	8
Register	USINT	9
Register	SINT	10
Register	UINT	2
Register	INT	1
Register	UDINT	4
Register	DINT	3
-	ULINT	27
-	LINT	26
Register	REAL	5
Register	LREAL	6
Register	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.

7.4.1 XML import of variables from another zenon project

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.

`IMPORT DBF FILE`

To start the import:

1. right-click on the variable list
2. in the drop-down menu 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.

Info

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 menu 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.

Info

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) when exporting.

DBF files must:

- correspond to the 8.3 DOS format for filenames (8 alphanumeric characters for name, 3 character suffix, no spaces)
- Be stored close to the root directory (Root)

DESIGN

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 "KANALNAME" (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)
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 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 (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises KANALTYP and DATENART
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	Resource 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	Linked 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 to 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	Hysteresis in %
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	Function linking
ASK_FUNC1	R	1	With interrogation before execution
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 Driver variables

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

- Information
- Configuration
- Statistics and
- Error messages

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.

Hint: 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.

Info

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 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 stopped
LineStateUnknow	BOOL	24.15	Connection status unknown
ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped

			For `driver stop`, the variable has the value `TRUE` and an **OFF** bit. After the driver has started, the variable has the value `FALSE` 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 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.
Baud rate	UDINT	10	Baud rate of the serial interface.
Parity	SINT	11	Parity of the serial interface
ByteSize	SINT	14	Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection.
StopBit	SINT	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	UDINT	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 MESSAGES

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	UDINT	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	SINT	25	Error message as number
DrvErrorMsg	STRING	30	Error message as text
ErrorFile	STRING	15	Name of error log file

8. Driver-specific functions

This driver supports the following functions:

BLOCKWRITE

Blockwrite allows for the efficient sending of multiple set values (e.g. recipes). Variables that lie next to each other in the PLC memory will be written to with a single write telegram or combined into a few telegrams (for larger areas).

Attention: If blockwrite is activated, the write sequence of the variables does not necessarily have to match their sending sequence.

Blockwrite can be activated with an entry in the project.ini file:

1. select the project in Project Manager
2. press shortcut `Ctrl+Alt+E`
3. the SQL folder of zenon opens in the Windows Explorer
4. `C:\ProgramData\COPA-DATA\[SQL folder]\[UID]}FILES`
5. navigate to `\zenon\system\`

-
- open the file project.ini with a text editor.
- add the following entry:

```
[MODBUS_ENERGY]
BLOCKWRITE=1
```

SEQUENCE OF EVENTS

The driver contains functions for reading out the event buffer of the PLCs mentioned in chapter MODBUS_ENERGY (on page 5). This functionality can be activated via the driver dialog Settings (on page 14). The type of PLC can only be set globally for the driver.

THE READING PROCEDURE

The MODBUS protocol makes it not possible for the slave to send a message without being contacted by the master first. The PLC must save spontaneous events and their exact time and can send this information to the master after the master made a request to the slave.

Sequence of Events (SOE) is a buffer with events (values and time stamp) which are stored in a special memory section in the PLCs. The address, the size , the format, and the procedure of reading from the buffer is manufacturer-specific as the MODBUS protocol does not have any guidelines for this.

The driver supports the spontaneous communication only for events which must be read out from the buffer in the sense that only these values receive a time stamp in the PLC. Thus they can be assigned in the same way as spontaneous events everywhere (AML, CEL, Historian) in the control system.

The driver checks the appearance of new events periodically with the highest priority used in the driver configuration.

THE INITIAL STATUS OF THE EVENT VARIABLE

When connecting to the PLC the driver gets the initial status for events from the event buffer (PLC-specific). If an event equals a distinct address in the MODBUS register and it was engineered as `SOE register event` , it is possible for the driver to adjust the possible initial values with the respective register. If however the initial status of the event variable was read out from the register (and not from the event buffer) and the original time stamp of the event is not available in the PLC anymore, this variable receives the local PC time as time stamp and receives the status bit T_INTERN `(real time internal)`.

AREVA MICOM P125/P126/P127

After the communication has started the driver fetches all events which are available in the PLC from the buffer (Register Page 35H). After that the retrieval (with acknowledgment) of the saved event recording of the Page 36H takes place.

For register events which are not available in the buffer the initial value - if available on Page 0H - is read out from the respective MODBUS register and stamped with the local PC time.

`Numbered Events` which are not available in the buffer receive initial value 0 and are stamped with the local PC time.

GE MULTILIN F650

After the communication has started the driver fetches the initial status of the events from the register (address 0xF000). After that the retrievals from the ring buffer (0xFD00) take place.

The events reported by the PLC arrive with the time stamp of the PLC and also receive the status bit T_EXTERN `(real time external)`.

SCHNEIDER SEPAM

Events are created in area "SOE Numbered Events". As offset the coil bit address of the setting is stated which should be read spontaneously. Supported are addresses 1000h - 105Fh. The read out of the initial state after a new connection is supported.

THE DRIVER OBJECT TYPE OF THE EVENTS

For the events two `Driver object types` are possible: `SOE - Numbered event` and `SOE - Register event` whereas not both have a function for each PLC.

AREVA MICOM P125/P126/P127

For Areva MiCOM both object types - `Numbered event` and `Register event` - are supported.

The `SOE - Numbered event` matches an event which is only generated if a condition occurs (and not if a condition disappears). The identification of an event is defined with the help of the `Offset` setting in the variable addressing (e.g. offset 5 for event 05). If an event occurs in the PLC, the associated BOOL variable is set to 1 and immediately back to 0 by the driver, The coming values by evaluated by applying ALM/CEL/Historian or a function for limit violation.

The `Register events` match the rest of the events. The `Register events` are assigned to one or several bits in a special register of the Page 0 in the AREVA PLC (e.g. offset 20h, bit 0). This means that the PLC makes sure that the event reflects the change of a status in a register. In the variable addressing the identification of the event takes place via `Offset` and `Bit` and describes the associated data points in a Page 0 register. You can use the datatypes BOOL, USINT and UINT.

Consult the documentation of the AREVA PLC to find out about the event types and which events are supported by the AREVA PLC.

GE MULTILIN F650

For GE Multilin F650 only object type `SOE - Numbered event` is supported.

Via the driver object type`SOE - Numbered event`, you can assign a BOOL variable to every event. The identification of the event is defined via the `Offset` setting of the variables (0 - 191).

The driver object type `SOE - Register event` has no function for the GE Multilin F650 PLC.

8.1 IEC NPx800

The reading of the events is only supported for the simplified event mode of the NPx800: only On events or Wiper events but no Status events (on/off).

Event variables are created under the primitive type **`Numbered events`** as BOOL variables. The event parameter can be read by creating a string variable under the primitive type **`Numbered events`**.

Attention: The string variable only has one valid values when the event has just been triggered. Otherwise the value of the string variable is an empty string. It behaves in the same way as a BOOL variable , i.e. just like a Wiper event. The zenon AML/CEL entry can be triggered either by the BOOL variables or by the string variable. The display of the string is only possible via dynamic limit texts.

The event number is configured with the help of the offset.

SETTING VALUES

The process of writing the set value when an event occurs:

1. The parameters are written in a possibly existing string variable.
2. The possibly existing BOOL variable is set to 1.
3. The possibly existing BOOL variable is set to 0.
4. The length of the possibly existing string variable with the parameters is set to 0.

PARAMETER STRING

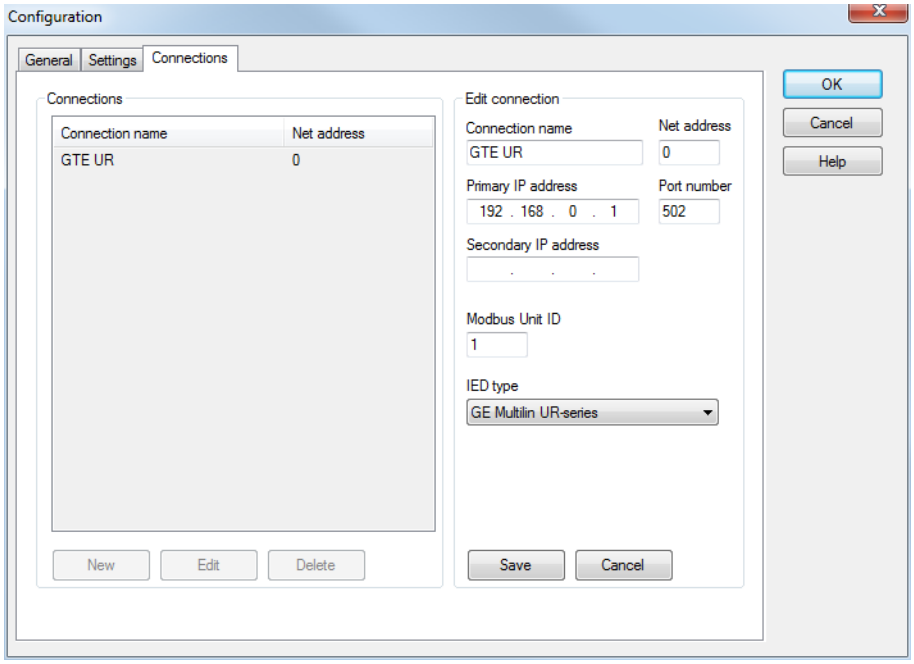
The structure of the parameter string (integer values separated by semicolons):

- WORD 0: Event data at word offset 2
- WORD 1: Event data at word offset 7 (parameter 0)
- WORD 2: Event data at word offset 8 (parameter 1)
- WORD 3: Event data at word offset 9 (parameter 2)
- WORD 4: Event data at word offset 10 (parameter 3)
- WORD 5: Event data at word offset 11 (parameter 4)
- WORD 6: Event data at word offset 12 (parameter 5)

- WORD 7: Event data at word offset 13 (parameter 6)
- WORD 8: Event data at word offset 14

8.2 GE Multilin UR series

The IEC type is selected in the Connections (on page 17) tab of the driver configuration:



Click on the drop-down list in the **IEC type** section to open the selection of the IEC types. Select **GE Multilin UR series** there.

SOE SUPPORT

Via the SOE functionality of the driver events can be read from the device. For this file **EVT.TXT** is analyzed at the device. Events must be created of driver object type SOE - Numbered event and of data type BOOL. The addressing takes places via the offset setting.

A list of event numbers and their meaning can be called via the web server of the UR series. **http://IP_Address_ of_UR_unit/FlexOperandStates.htm** (z.B.: **http://192.168.37.67/FlexOperandStates.htm**).

The SOE - Numbered event matches an event which is only generated if a condition occurs (and not if a condition disappears). The identification of an event is defined with the help of the Offset

setting in the variable addressing (e.g. offset 5 for event cause 5). If an event occurs in the PLC, the associated BOOL variable is set to 1 and immediately back to 0 by the driver, The coming values by evaluated by applying ALM/CEL/Historian or a function for limit violation.

FILE TRANSFER

Via a command variable of the driver object type **`File transfer`** (see file transfer description for ICE devices) the file is transferred to the local computer. The command consists of: **`GET`** followed by a blank space and the file name: **`GET EVT.TXT`**

The file is copied to the file transfer folder (as defined on tab Settings (on page 14) of the configuration). Format of the file name: `Net address+underscore+file name.`
The net address is taken over from the driver/variable and is only used for saving so that the data are not overwritten for several controls.

Example: `0_EVT.TXT`

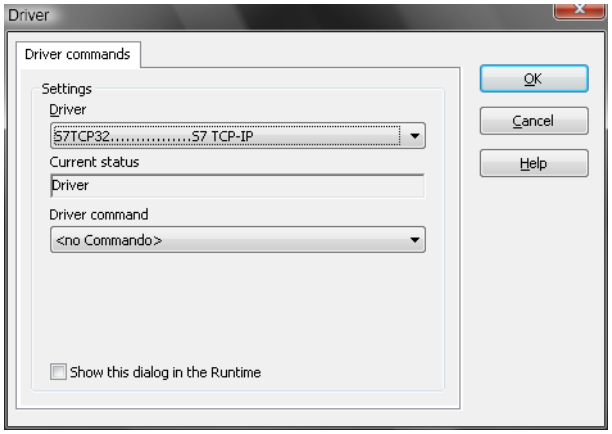
9. Driver commands

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 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 set value	Write set value to a driver is allowed.
▸ Deactivate driver	Write set value to a driver is prohibited.

write set value	
▸ 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 7.00 -> 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 \zenon700\LOG for zenon version 7.00 SP0. 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.

10.2 Check list

Checks after communication errors:

- Is the PLC connected to the power supply?
- Are the participants available in the **TCP/IP** network?
- Can the PLC be reached via the **Ping** command?
- Can the PLC be reached at the respective port via **TELNET**?
- Are the PLC and the PC connected with the right cable?
- Did you configure the net address correctly, both in the driver dialog and in the address properties of the variables?
- Did you use the right object type for the variable?
- Does the offset addressing of the variable match the one in the PLC?
- Analysis with the Diagnosis Viewer: Which messages are displayed?