zenon manual

Variables

v.7.10

COPA·DATA
do it your way

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. Variables

Variables, also called process variables or data points, are the interface between the data source (PLC, field bus, etc.) and zenon. They represent certain measured values or states of the hardware, including properties such as scaling, limit values, etc.

The variables are defined in the central variable list of a project and are available here from everywhere (functions, screens, archives, etc.). Integration projects can directly access the variables of lower level projects.

A variable is always based (on page 72) on two components: data type (on page 12) and driver object type (cti.chm::/28685.htm). These two components are independent and thus can be configured independently.

📘 **License information**

Part of the standard license of the Editor and Runtime.

PROJECT MANAGER CONTEXT MENU

Menu item	Action
New variable...	Opens the wizard for creating a new variable.
Export XML all...	Exports all entries as an XML file.
Import XML...	Imports entries from an XML file.
Extended import/export	Opens a drop-down list with pre-defined import and export formats.
Display unused variables	Creates a project analysis for unused variables in the current project and displays it as a result list in its own window.
Open in new window.	Opens a new window in order to view and edit the variable. (Default: at the bottom of the Editor.)
Editor profile	Opens the drop-down list which includes pre-defined Editor profiles.
Help	Opens the online help.

Each variable can have its own attributes. A total of 64 statuses/attributes have been defined.

Info

Screen elements that are linked to a variable that have neither a value nor a status are switched to invisible in Runtime.

2.1 Variables detail view of toolbar and context menu

TOOLBAR

no.	Symbol	Action
01	New `variable`	Opens the dialog for creating a new variable.
02	Copy	Copies selected entries to the clipboard.
03	Paste	Pastes the contents of the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of".
04	Delete	Deletes selected entries.
05	Create standard function	Opens the wizard for selecting the variables and the set values and creates a matching function. The action is documented in the output window.
06	Edit selected cell	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line.
07	Replace text in selected column	Opens the dialog for searching and replacing texts.
08	Replace text in limits	Opens the dialog for selecting dynamic elements and after that the dialog for searching and replacing.
09	Properties	Opens the `Properties` window for the selected entry.
10	Expand all	Drop-down list in order to expand or to collapse all nodes or the selected nodes.
11	Activate	Activates the selected elements of a structure variable.
12	Deactivate	Deactivates the selected elements of a structure variable.
13	Activate all	Activates all inactive elements of a structure variable.
14	Export selected XML	Exports selected entries as an XML file.
15	Import XML	Imports XML files.
16	Import S7 project	Imports a Simatic S7 project.
17	Activate	Activates or deactivates the extended filter settings.
18	Jump back to starting element	If you entered the list via function `linked elements`, the symbol leads back to the start element. Only available in the context menu when all linked elements are opened.
19	Remove filter	Removes all filter settings.
20	Help	Opens the online help.

CONTEXT MENU

Menu item	Action
`Variable new`	Opens the dialog for creating a new variable.
`Copy`	Copies selected entries to the clipboard.
`Paste`	Pastes the contents of the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of".
`Delete`	Deletes selected entries.
`Create standard function`	Opens the dialog for selecting variables and set values.
`Edit selected cell`	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line.
`Replace text in selected column`	Opens the dialog for searching and replacing texts.
`Replace text in limits`	Opens the dialog for selecting dynamic elements and after that the dialog for searching and replacing.
`Properties`	Opens the `Properties` window for the selected entry.
`Expand/reduce nodes`	Opens a menu with which you can expand or collapse the nodes.
`Activate all`	Activates all inactive elements of a structure variable.
`Activate`	Activates the selected elements of a structure variable.
`Deactivate`	Deactivates the selected elements of a structure variable.
`Export selected XML`	Exports selected entries as an XML file.
`Import XML`	Imports from an XML file.
`Extended import/export`	Opens the menu for importing and exporting S7 projects, dBase and CSV.
`Variable use`	Creates a project analysis for selected variables in the current project and displays it as a result list in its own window.
`Extended filter`	Opens a menu for activating or deactivating filters or for removing all filter settings.
`Linked elements`	Shows the elements which are linked to the variable.
`Help`	Opens the online help.

2.2   Naming of objects

The name of an object must be unique in zenon.

> 💡 **Info**
>
> *Note the following when assigning names:*
>
> ▸ The **#** character is not permitted for variable names.
>
> ▸ The maximum length for `Name, Identification` and `String archive filler value` is 128 characters each.
>
> ▸ The maximum length for `Resources label` is 255 characters.

The variable names are case-sensitive. You can create variables with the same name which only differ from each other by writing a single letter in upper or in lower case. For example the variables `test` and `Test` are two different variables.

Attention: Identical names, which only differ from each other due to capitalization, can be a problem with drivers or zenon Logic with:

1.   Driver configuration:

 If the driver name only differs in terms of capitalization, both drivers use the same allocation file.

2.   zenon Logic:

 zenon Logic does not differentiate between upper and lower case. Two variables which differ in zenon only by case sensitivity cannot be used in zenon Logic. This concerns variables:

- for a zenon Logic project with zenon Logic driver, IEC870 driver and IEC850 driver
- which are set to `externally visible` in zenon
- which are assigned to a driver for which a zenon Logic project for `simulation - programmed` has been created

> 💡 **Info**
>
> *Recommendation: Use unique object names which differ from each other not only by case sensitivity.*

2.3   Activating variables in zenon

Variables are activated in zenon when screens to which they are linked are opened. They are read from this point onwards. When closing a variable, the linked variables are deactivated again and reading is ended.

ACTIVATE (ADVICE):

Variable values are always queried by the driver. This happens regardless of whether they are needed at the moment in the project or not, even if, for example, a variable has no limit value, is not displayed and the value is also not recorded. The driver thus always has current values and the current status of the variables and can provide these if necessary without querying them.

- ▶ Advantage: Data is immediately available. For example, the toggling of variable values in the `Write set value` Function (on page 170) or in screen elements such as **Switch** is executed more quickly because the variable does not need to be queried especially before switching.

- ▶ Disadvantage: Higher load for the communication to the control unit.

The following functionalities ensure that the variables are always activated:

- ▶ Alarms administration

- ▶ CEL

- ▶ Archive Manager

- ▶ VBA/VSTA Online Container

- ▶ Property`Harddisk data storage active` activated

- ▶ Property`DDE active` activated

- ▶ `Keep update list in the memory` option is activated in the driver configuration (**General** tab)

Variables that are activated via the `DDE active` property are also available to other Windows applications.

Hint: If the variable value is to be toggled, we recommend activating the `DDE active` property.

QUERIES (REQUEST):

If variables are not activated, values and status are queried from the driver if necessary.

If variable values need to be asked especially, Runtime waits until the driver knows the value of the variables. If the value is not available due to a breakdown in communication with the control unit, the waiting time corresponds to the timeout time of the driver.

2.4 Datatypes

Each variable is based on an IEC data type. The data type has the same properties as the variable itself (unit, signal resolution, limits, etc.). This does not include driver-specific properties such as addressing of the PLC, for example. A data type is a variable template without connection to the process. This connection to the process is only established with the driver object type (on page 31) for the variable.

On creating a new variable a data type has to be selected. All properties of the data type are inherited to the variable. A reference to the properties of the data type is established, i.e. if a property of the data type is changed, it is also changed in all variables linked to that data type.

If a property is a reference, it is marked with a symbol in the properties window.

This reference can also be cut. A more detailed description can be found in the Inheritance concept (on page 106) chapter.

CONTEXT MENU PROJECT MANAGER

Menu item	Action
New simple datatype...	Opens the dialog for creating a new data type.
New structure datatype...	Opens the dialog for creating a new structure data type.
Export XML all...	Exports all entries as an XML file.
Import XML...	Imports entries from an XML file.
Help	Opens online help.

IEC DATATYPES:

IEC data types are standardized in the IEC 61131-3 by the IEC. At the the moment zenon supports the following IEC data types:

Short name	Long name	Comment / value range	Number of bits
BOOL	Boolean	Bit: 0/1	1
BYTE			
SINT	Short integer	Signed byte: -128 to 127	8
USINT	Unsigned short integer	Byte: 0 to 255	8
INT	Integer	Signed word: -32768 to 32767	16
UINT	Unsigned integer	Word: 0 to 65535	16
DINT	Double integer	Signed double: -2147483648 to 2147483647	32
UDINT	Unsigned double integer	Double: 0 to 4294967295	32
LINT	Long Integer	Because zenon cannot hold the value range of a 64 bit number, the actual value range is restricted to 52 bits. Numbers with signs are possible from –2251799813685248 to 2251799813685247. Numbers outside of this range cause a overflow or underflow, according to a 52-bit integer. The following components work with a full 64 bit resolution in the zenon Runtime: 1. The shared memory VBA Interface, when the value was entered in signal resolution. 2. The shared memory zenon Logic interface. 3. The driver kit works for the whole range of 64 bits, and so do all the drivers, provided the	64 or 52

		driver supports this IEC data type. At the moment, these are **Sample32, Internal** and **STRATON32.** RGM and recipes cannot save even a 52 bit value with full accuracy.	

ULINT	Unsigned long integer	Because zenon cannot hold the value range of a 64 bit number, the actual value range is restricted to 52 bits. Numbers without signs are possible from 0 to 4503599627370496. Numbers outside of this range cause a overflow or underflow, according to a 52-bit integer. The following components work with a full 64 bit resolution in the zenon Runtime: 1. The shared memory VBA Interface, when the value was entered in signal resolution. 2. The shared memory zenon Logic interface. 3. The driver kit works for the whole range of 64 bits, and so do all the drivers, provided the driver supports this IEC data type. At the moment, these are Sample32, Internal and STRATON32. RGM and recipes cannot save even a 52 bit value with full accuracy.	64 or 52
REAL	Real numbers	Real numbers	32
LREAL	Real numbers	Real numbers	64
STRING	Variable-length single byte character string	ASCII string (the max. string length depends on the driver). The string length is set to 5 characters by default. The correct length is to be given during configuration if necessary.	>= 8 x string length
WSTRING	Variable length multi-byte character string	Contains multi-byte strings, for example strings in Unicode coding (UTF-8 etc). The string length is set to 5 characters by default. The correct length is to be given during configuration if necessary.	>= N x string length

DATE	Date	IEC date in steps of 1 day	16
TIME	Duration	Duration in the IEC format. IEC time in steps of 1 ms, signed integer	32
TOD	Time of day	Time of day in steps of 1 ms	32
DATE_AND_TIME	Date and time of day	Defines a period of time with 64 bit and is saved in a binary-coded decimal format.	64

⚠ **Attention**

In zenon the data types DATE, TIME, TOD are treated as a data type UDINT in the Unix time format (bygone seconds since 01.01.1970). The data type DATE_AND_TIME equals data type FLAOT in the Unix time format. With this the time is saved exactly to millisecond, i.e. three digits after the decimal point. The driver coverts the data type used in zenon to the corresponding data type and sends the new value to the PLC.
Not all drivers support these data types. Please check the corresponding driver documentation whether the used driver supports the data types.

You can find information on how the inheritance of data type properties works in the Inheritance concept (on page 106) section.

2.4.1   Data types detail view toolbar and context menu

CONTEXT MENU

Menu item	Action
`New simple data type`	Opens the dialog for creating a new simple data type.
`New structure data type`	Opens the dialog for creating a new structure data type.
`New structure element`	Opens the dialog for adding a structure element to a structure data type.
`Copy`	Copies selected entries to the clipboard.
`Paste`	Pastes the contents of the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of".
`Delete`	Deletes selected entries.
`Edit selected cell`	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line.
`Replace text in selected column`	Opens the dialog for searching and replacing texts.
`Replace text in limits`	Opens the dialog for selecting dynamic elements and after that the dialog for searching and replacing.
`Properties`	Opens the `Properties` window for the selected entry.
`Export selected XML`	Exports selected entries as an XML file.
`Import XML`	Imports XML files.
`Extended import/export`	Open the menu for importing S7 projects, dBase and CSV.
`Remove all filters`	Removes all filter settings.
`Help`	Opens the online help.

2.4.2 Types of data types

Basically there a two different kinds of data types:

▶ simple data types; in turn, these consist of:

- pre-configured simple data types (on page 18)
- user-defined simple data types (on page 18)

▶ Structure data types (on page 22)

Pre-configured simple data types

These are delivered together with zenon and are immediately available when a new variable is created. They conform to the standardized `simple data types` of IEC 61131-3 such as INT, USINT etc. They define a set value area (fixed upper and lower limit) with a defined number of values. That is why real numbers can be depicted as float-point numbers only with a certain accuracy.

The properties of the `simple data types` can be changed. The names of these data types always match with the names of the basic IEC data types, these and the IEC data types cannot be deleted or renamed.

The data types only become visible in zenon, once a driver is created. Each driver opens a list of data types it supports. If an IEC data type does not exist in the list of data types, it is not supported by any driver in the project.

Changes in the properties of the data types affect all linked variables and thus also affect all linked structure data types.

User-defined simple data types

These data types can be created by the user. On creating an existing data type is used. All properties of this basic data type are `copied` to the new data type and can be changed later on. There is no reference to the data type on which it was originally based. All data types can therefore be configured independently of one another.

For the user-defined data types all properties including name and IEC data type can be changed.

⚠ **Attention**

All variables that are based on a data type where the IEC data type is changed may need to be subsequently configured manually.

Example

An IEC data type INT is changed to an IEC data type BOOL. All set value limits of all variables and at all positions, where the variable is used (screens), are no longer correct. It becomes even more critical, if the properties in the variables were overwritten and no longer have a reference to the data type!

When creating a new user-defined data type, a basic data type that has the same IEC data type to that which is needed should always be used.

> ⚠ **Attention**
>
> *A subsequent change of the IEC data type explicitly is not recommended!*
>
> *A number of settings depend on the IEC data type and have to be re-configured by hand if the IEC data type is changed. (Example: Signal resolution, measuring range, hysteresis, set value limits, non-linear value adjustment, etc.).*
>
> *When changing the IEC data type, all properties specific to this data type have to be checked! Sometimes, the properties have to be adapted* `before` *the change, since for BOOL variables the signal resolution and the measuring range cannot be changed, for example.*

Creating a new user-defined datatype

Open the folder Variables and select `Datatypes`.

Now all existing datatypes are listed in the detail view. On the first opening only the pre-defined datatypes corresponding to the IEC datatypes are in this list. A user-defined datatype is always based on

an IEC datatype. Click into the list with the right mouse button an in the context menu select `Create simple datatype....`

In the dialog opening now enter the name of the user-defined datatype and select an existing datatype. This datatype is a template for the new datatype to be created. All properties are copied into the new one. They can subsequently be changed in the properties window. No reference to the basic datatype, being the basis of the new one, exists any longer.

💡 **Info**

The name of the data type is limited to 128 characters.

Click on `Finish`. The new user-defined datatype now appears in the list of simple datatypes in the detail view.

Changing the properties of a user-defined data type

The properties of a user-defined data type can be changed in the properties window, where all properties of the data type are listed.

Please pay attention that you do not modify the IEC data type because thereby the signal resolution would also be modified.

Deleting an user-defined datatype

Select the datatype to be deleted in the list in the detail view and confirm the delete action in the following dialog box.

⚠ **Attention**

If an user-defined datatype is deleted, all variables based on this datatype are also deleted.

Structure data types

Structure data types are always user-defined data types. In contrast to the simple data types that only allow a flat variable list, they allow to build a structure. This structure can even be nested and so it can become very complex.

If a structure variable is needed, a structure data type has to be created first. A structure consists of a structure name and structure elements. Structure elements can be simple data types or other structures. Structure elements can also be arrays (fields) with up to three dimensions.

⚠ **Attention**

Structure variables can only be newly created. Changes afterwards are not possible.

Arrays can only be newly created. Changes afterwards are not possible.

You must define, during creation, whether the structure element should use a `linked` or `embedded` type. Changes afterwards are not possible.

EXAMPLE FOR A STRUCTURE DATATYPE:

Name △	Data type	Description
Filter text	Filter text	Filter text
Voltage	INT	Simple data type
⊟ Structure Controller	0	Structure data type
Engine speed set	INT/<embedded2>	Structure element
Engine speed actual	INT	Structure element
⊟ Structure Engine	0	Structure data type
Activity Input	UINT	Structure element
Charging Rate	INT/<embedded1>	Structure element
Voltage	Voltage	Structure element
Temperature[3]	SINT	Structure element
⊟ Engine speed control	Structure Controller	Structure element
Engine speed set	INT/<embedded2>	Structure element
Engine speed actual	INT	Structure element

Two structures are displayed in the illustration above.

The structure Controller consists of two elements: `Engine speed set` and `Engine speed actual`.

▸ The structure element `Engine speed actual` is `linked` to the datatype INT, i.e. all properties of the datatype INT are inherited to this structure element.

▸ The structure element `Engine speed set` is `embedded` into the structure. So all properties of this datatype can be configured in the structure element itself.

The structure Engine consists of the following structure elements: `Activity Input`, `Charging Rate`, `Voltage`, `Temperature[3]` and `Engine Speed Control`.

▸ The structure element `Activity Input` is linked to the pre-defined data type UINT - all properties are taken from UINT.

▸ The structure element `Charging Rate` has the embedded data type INT - all properties can be defined in the structure element.

▸ The structure element `Voltage` is linked to the user-defined data type Voltage. The element gets all properties from this data type.

▸ The structure element `Temperature` is a one-dimensional array of three elements of the pre-defined data type SINT.

▸ The structure element `Engine Speed Control` is linked to the structure Controller mentioned above. Structures in a structure cannot be embedded. They are always linked to the original structure. The structure contains two further structure elements: Engine Speed Set and Engine Speed Actual, which get their properties from the original structure.

Creating a structure datatype

This is how you create a structure datatype:

1. Right click on a user-defined datatype in the list and select `New structure datatype` in the context menu.

2. You define the names in the dialog box that now appears.

Note: The name of the data type is limited to 128 characters.

3. Confirm the settings with `Finish`. The name of the structure datatype can be changed later on in the properties window.

4. Now a dialog opens, with which the first structure element can be created.

5. Give it a clear name.

6. Select the data type upon which the element should be based. Click on the ... button to open the selection dialog.

You can also link structures as structure elements. It is not possible to embed structures as structure elements. It is also only possible to link entire structures and not parts of a structure.

7. Define if the data type is to be embedded or linked.

8. Define the array dimensions.

    If you do not want to use an array, set the dimensions to 0.

9. Confirm the settings with **Finish**.

EMBED OR LINK

Parameters	Description
Embedded structure elements	can have properties that differ from their basic datatypes. The properties are defined individually for each structure element and are saved there.
Linked structure elements	Always get all their properties from the basic datatypes that they are linked to. If the basic data type is changed, all linked elements are changed in the same way!

Inserting further structure elements

To insert further structure elements:

1. Right-click on the desired structure data type.

2. Select `New structure element...` in the context menu.

3. Define it in the dialog box that opens

   a) A clear description (name is limited to 128 characters)

   b) the array dimensions and

   c) whether you wish to embed or link the structure element.

If you now highlight the structure data type in the detail view, the existing structure elements are shown as sub-points.

Changing the properties of a structure element

To change the properties of a structure element:

EMBEDDED STRUCTURE ELEMENTS

1. Highlight the structure element in the detail view.

2. Change the desired properties in the properties window

LINKED STRUCTURE ELEMENT

With linked structure elements, the following can be changed in the Properties window:

- ▶ Name

- ▶ Offsets

- ▶ Array dimensions

- ▶ Description

All other properties must be changed directly at the original data type.

> ⚠ **Attention**
>
> *Note that changes to the data type have a direct effect on all variables related to them and also have an effect on all other structure data types linked to this data type.*
>
> *Changes to the data type are not recommended. These must be prepared and carried out with due care. In particular when changing the IEC data type settings, measuring range settings, set value limit settings, etc.*
>
> *In certain circumstances, all variables and all screens in which these variables are used must be manually adjusted afterwards!*

Moving a structure element

Elements of a structure data type can be moved by dragging & dropping. To move elements:

1. Highlight the desired element.

2. You can also select several interrelated elements

3. Click in the selection and drag the element to the desired position

For multi-user projects, a check is carried out to see if the status of the data type to be changed is set to `Allow changes` or can be set to this.

MOVING SEVERAL OBJECTS

interrelated elements can be moved by dragging & dropping. For this, the following applies:

▶ Only interrelated structure elements can be moved.

MALUGG_STRUCT	Struktur-Datentyp
IN_1	Struktur-Element
IN_2	Struktur-Element
IN_3	Struktur-Element
IN_4	Struktur-Element
IN_5	Struktur-Element
IN_6	Struktur-Element
IN_7	Struktur-Element

▶ If a gap is recognized with multiple selection, the drop action is refused.

MALUGG_STRUCT	Struktur-Datentyp
IN_1	Struktur-Element
IN_2	Struktur-Element
IN_3	Struktur-Element
IN_4	Struktur-Element
IN_5	Struktur-Element
IN_6	Struktur-Element
IN_7	Struktur-Element

Deleting a structure datatype

To delete a structure data type:

1. Right-click on the structure data type in the list.

2. Select `Delete` in the context menu.

3. The structure data type is deleted with all its structure elements.

> ⚠ **Attention**
>
> *When being deleted, all variables based on this structure data type are deleted!*

2.5   Driver

zenon offers more than 300 different connections to different PLCs, bus systems and applications.

To communicate with a data source it is necessary to link an interface driver (protocol driver). The driver establishes the connection between a PLC and zenon. The data source not necessarily is a PLC. The DDE driver communicates with a DDE server, the OPC client driver with an OPC server, the SNMP driver with SNMP agents, etc.

A number of different drivers are available and they also can be used at the same time.

> ⚠ **Attention**
>
> *Under Windows CE only one type of a driver can be started at the time.*

CONTEXT MENU PROJECT MANAGER

Menu item	Action
New driver...	Opens the detail view for selecting a driver.
`Help`	Opens online help.

On creating a new variable, it has to be defined for which driver the variable is created.

The zenon drivers are protocol drivers. They should not be confused with the interface drivers of the operating system (e.g. drivers for LAN cards of drivers for a serial interface). The zenon drivers are always based on the interface drivers of the operating system and communicate with the respective protocol (e.g. Modbus RTU, MPI, Melsec A, etc.) that is understandable for the PLC.

Basic drivers are available in zenon free of charge: Variables based on these drivers are not counted for licensed I/Os:

Parameter	Description
`Simulator driver` SIMUL32 (Main.chm::/SIMUL32.chm::/SIMUL32.htm)	For internal variables. These can be defined as failure-proof variables (hard disk data) or as flag types; flag objects can automatically change their value for dynamic simulation
`Mathematics driver` MATHDR32 (Main.chm::/MATHDR32.chm::/MATHDR32.htm)	Variables of this driver are used for the calculation of mathematical functions or for counters.
`System driver` Sysdrv (Main.chm::/Sysdrv.chm::/Sysdrv.htm)	variables for monitoring and controlling the hardware, the network and other project-specific properties.
`Internal driver` Internal (Main.chm::/Intern.chm::/Intern.htm)	User-defined variables without connection to a PLC. (Similar to the user-defined variables of the system driver). Each variable has an individual value. So it is not possible e.g. to extract bit or byte variables from an integer variable. Defining address information is not necessary, the according properties in the properties window cannot be edited.

Detailed driver descriptions for the individual drivers can be found in the corresponding driver documentation, which is available to you via the online help in the Driver chapter and on the installation DVD.

When a new driver is created, a selection dialog is opened, where all available drivers are listed. The information for this dialog are in the files `Treiber_DE.xml`, `Treiber_EN.xml` etc. depending on the language. You can open or edit these XML files with the `driverinfo.exe.` tool

2.5.1   Driver detail view toolbar and context menu

CONTEXT MENU

Menu item	Action
`Driver new`	Opens the dialog for creating a new driver.
`Change driver`	Opens the selection window for defining a driver.
`Delete`	Deletes selected entries after a confirmation from the list.
`Driver configuration`	Opens the dialog for configuring the driver.
`Import variables from the driver`	Opens the dialog for importing variables.
`Edit selected cell`	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only available if editing is possible.
`Replace text in selected column`	Opens a dialog in order to search for characters in the column on which the focus is and replace them automatically.
`Properties`	Opens the `Properties` window for the selected entry.
`Remove all filters`	Removes all active filters from the list.
`Help`	Opens the online help.

2.5.2   Driver object type

The driver object types define which area of a variable should be referred to in the PLC. At the moment all manufacturer use different names for the areas of their PLCs. In the Siemens world there are datablocks, markers, in/output etc. in the Modbus world there are coils and holding registers. Each driver creates an internal list with the available driver object types. When creating a new variable, the area in the PLC to which it is allocated must be specified.

THERE IS A STRICT DISTINCTION BETWEEN DATA TYPE AND DRIVER OBJECT TYPE

The strict distinction makes it possible to create a structure as a data type completely independently and thus independently of the PLC. This structure as a variable can be linked to any driver. A structure

can, for example, be linked to the corresponding driver 1:1 with a Siemens PLC as well as a Modbus PLC with the corresponding drivers.

2.5.3 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.

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.

2.5.4 Configuration of a driver

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.

2.5.5   Driver simulation

If the underlying process is not available when configuring, this can be simulated and tested in advance. Three modes are available for this:

- ▶   simulation - static (on page 38): constant values simulated by the driver

- ▶   simulation - counting (on page 38): values simulated by the driver are counted up

- ▶   simulation - programmed (on page 39): Values are calculated via a simulation project with zenon Logic

> ⚠ **Attention**
>
> *If the driver is stopped in mode* **`Simulation - counting`**, *only the counting is stopped. The variable is not switched to* `faulty`. *In all other modes the driver is really stopped.*

Hint: **`Simulation - programmed`** is not supported by drivers for:

▸   Internal variables

▸   Mathematical variables

▸   Simulator variables

▶ System variables

> 💡 **Info**
>
> *zenon variables which represent the zenon Logic IO variables are available in the project in state* `Driver simulation programmed`.

> 📄 **License information**
>
> *Part of the standard license of the Editor and Runtime.*

Simulation static

For a static simulation, no communication to the control is established; the values are simulated by the driver. In this mode the value remains constant. Values can be changed by the Runtime or the user. At a restart of the Runtime with `Simulation - static` theses values will however not be saved and are lost.

Simulation - counting

For a counting simulation, no communication to the control is established; instead the values are simulated by the driver. In this mode, the driver increments the values within a value range automatically, starting with 0. If the maximum value has been reached, the counting process starts at 0 again.

> 💡 **Info**
>
> *For negative starting values, the counting process only starts at 0.*
>
> *INT uses the maximum value from USINT to count.*

> ⚠️ **Attention**
>
> *If the driver is stopped in mode* `Simulation - counting`, *only the counting is stopped. The variable is not switched to* `faulty`. *In all other modes the driver is really stopped.*

Simulation - programmed

For Simulation - programmed, no communication is established to the PLC; instead, the values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime. It enables the status and time stamp of variables to be modified on the driver. Write commands to variables by zenon are forwarded to the simulation, redundant servers and standby servers are synchronized. This is how you also simulate complex processes.

To start the simulation program, select the Simulation - programmed mode in driver configuration (on page 45).

💡 Info

Windows CE

> *Simulation - programmed* is not available for Windows CE.

Editor

Create project

To create a program for Simulation - programmed, you must:

▶ select a single process driver

▶ make sure that the project name is valid.

> ⚠ **Attention**
>
> **Simulation - programmed** is not supported by drivers for:
>
> ▸ Internal variables
>
> ▸ Mathematical variables
>
> ▸ Simulator variables
>
> ▸ System variables

To create a simulation project:

▶ click in the group `Driver simulation project` in the property `Edit` on Click here ->

▶ A new zenon Logic project is created.

▶ `Name` and port numbers for `Event port` and `Standard port` are automatically issued; change these as you wish

▶ the zenon Logic Workbench is opened

The simulation project's workshop is automatically closed if:

▶ the zenon editor is closed

▶ a driver is deleted and its simulation project is currently being edited in Workbench

▶ the simulation project for the driver is deleted via the `Delete` property

▶ the simulation project is renamed

PROJECT ARCHIVING

A zenon Logic project contains many files and folders. To manage them, in particular to simplify distributed engineering, all files are archived in compressed form in the **Simul_<Treiber-ID>.zip** file in the `<Sql Projekt Pfad>\FILES\zenon\custom\drivers`. This file is in the driver files in the editor. When starting zenon Logic Workbench, the files are automatically decompressed are compressed again when it is ended.

CREATING VARIABLES

When creating variables in a simulation project, the succession is important:

1. create the variable

2. select the valid data type in the zenon Logic Workbench

3. after that activate property `embed symbol`

Background: Variables are created with the `PLC marker` driver object type by default. This object type does not support all data types for all drivers. If a variable is copied in the zenon Logic Workbench with active property `embed symbol`, you must deactivate this property in order to change the data type. With this the variable is then deleted in zenon.

> 💡 **Info**
>
> **Error message "Can not create variable"**
>
> *A variable cannot be created in the simulation object if the driver does not have driver object types - exception*
>
> ***driver variable*** *- which support this data type.*
>
> *On activation of* `embed symbol`*, the error message "can not create variable" is displayed.*

STRUCTURE DATA TYPES

At creating variable with a structure data type:

▸ they are created with the driver object type `PLC marker` if possible

▸ all driver object types with the exception of **driver variable** are tried until one can be used for all elements

▸ if no fitting driver object type is available,

- no variable is created for non-structure variables
- driver object type `PLC marker` is used for structure variables

EXAMPLE：

A structure data type contains elements of type UINT and STRING. A variable is created for S7TCP driver and **embed symbol** is activated.

▶ The variable is not created in `PLC marker` but as `Ext. Data module`, in which all structure elements are present.

▶ A new type LINT is added; it is not supported by the `Ext. Datablock`.

▶ When a new variable is created, it is created as type `PLC marker`. Only the first complex variable can be activated (UINT). The object type `Ext.` remains for the existing structure variable `Data` and the last structure element (LINT) cannot be activated.

COMPILING THE FILES

If a simulation project is compiled in zenon Logic Workbench, this causes a variable to be created in zenon and the Runtime program file.
When compiling, a subfolder is created in Runtime-Folder\RT\FILES\zenon\custom\drivers with the name of the simulation project. The file with the code of the simulation is SIMULRT.COD is archived in this folder.

The following message is displayed in the output window:

SIMULRT.COD

And if compiled with the C compiler, then also:

T5APP.DLL

> ⚠ **Attention**
>
> *For **multi-user projects** (on page 43) you must not create simulation projects offline. They cannot be deleted anymore.*

Delete project
To delete a simulation project:

1. click on the **Click Here->** button in the `Delete` property in the `Driver simulation project` group

2. confirm this when requested to do so

3. the ZIP file with the project files is deleted

4. the zenon Logic Workbench is ended

Note:

▶   the Runtime folder for the simulation project remains

▶   function `Undo` is not available for this action,

▶   click on `Edit` to create a new simulation project

▶   for `multi-user projects (on page 43):`

   •   the driver must be configured to `Make changes possible` so that the simulation program can be deleted

   •   the project is not displayed as deleted on other Clients as long as it has not been synchronized;

   if you try to open a deleted project before it has been synchronized (click on `Edit`), a new simulation project will be created

   •   you cannot reverse the deletion of single-user projects via `Cancel changes`

   > ⚠ **Attention**
   >
   > *The Simulation project is immediately removed from the local database and the server database when deleted. This action cannot be undone!*

Note: Manual deletion of the ZIP file of the driver also leads to the simulation project being deleted. Requirement: The zenon Logic Workbench for this driver is not opened. We do not recommend doing it this way!

Distributed engineering

For `multi-user projects`, all running zenon Logic Workbenches that are part of the project are closed when:

▶   `Accept changes` for modules that are completely locked for other users in `Enable changes,` such as variables, drivers and data types

▶   `Discard changes:`

   •   new projects are not created on the server

   •   Changes in the simulation program are lost

- Driver files not present in the server database are also lost in the local database

▶ Synchronize:

changes made locally to the simulation program program are lost

▶ Update local version:

changes made locally to the simulation program in the simulation program are lost

> ⚠ **Attention**
>
> *The status of the simulation project's ZIP file may not be additionally modified(`Accept`*
> *`Changes`, `Enable Changes`, `Discard Changes` multi-user status), to*
> *ensure correct `Accept Changes` and `Enables Changes` for the driver!*

DELETE PROJECT

For `multi-user projects`

▶ the driver must be configured to `Make changes possible` so that the simulation program can be deleted

▶ the project is not displayed as deleted on other Clients as long as it has not been synchronized;

- if you try to open a deleted project before it has been synchronized (click on `Edit`), a new simulation project will be created

▶ you cannot reverse the deletion of single-user projects via `Cancel changes`

> ⚠ **Attention**
>
> A simulation project with the status of `Enable Changes` cannot be deleted (on page 42) in `Offline` mode.
>
> A simulation project created `offline` in a `multi-user project` can therefore no longer be deleted.

Change driver

For a driver change in zenon the following is true:

▶ The simulation project is maintained.

▸ All ports are however set to 0. To receive working port numbers, open the project in the Editor. At this new port numbers are entered automatically. Port numbers can also be assigned manually.

> ⚠ **Attention**
>
> *Drivers that are linked to a zenon Logic project using the integrated solution cannot be exchanged.*

XML export/import

A project in mode `Simulation - programmed` *can be exported via XML. It can however not be imported as new project for another driver.*

Workaround:

1. Export the zenon Logic programs.

2. Import them to the new project for the new driver.

3. Replace the variables via search and replace.

Driver configuration

There are four modes available when configuring the driver:

▸ `Hardware`

▸ `Simulation - counting`

▸ `Simulation static`

▸ `Simulation - programmed`

Select the `simulation - programmed` mode.

Driver variable status

The status of zenon Logic is shown in the driver variables SimulRTState at Offset 60. The variable is numerical and cannot be written to. The value informs you of the status of Runtime:

Bit	Meaning	Description
31	without application	Runtime has not loaded a program, the program was stopped or not loaded to Runtime.
30	not instanced	Runtime was not instanced, for example because the DLL with Runtime could not be loaded, there is too little memory, the DLL is not present or is the wrong version.
18	p-code available	p-code is available together with compiled code, switching is possible.
17	compiled code active	Runtime runs with compiled code (otherwise: interpreted p-code)
16	compiled code available	Compiled code from the C compiler is available
15	application is loaded	An application is present, Runtime is running
14	can't start - missing handlers	Some "C" functions are missing
13	active breakpoints installed	At least one breakpoint was set by the debugger.
12	CT segment exists	Application was compiled with the "complex variables in separate segment".
11	Reserved	(internal, for Runtime only.)
10	sysinfo request is available	(internal, for Runtime only.)
9	freeze event production	Binding does not transfer any events.
8	single cycle mode	(intern, for Runtime and debugger only.)
7	Reserved	(internal, for Runtime only.)
6	locked variables	At least one variable is blocked.
5	trigo functions are in degrees	Trigonometric functions are stated in degrees.
4	log message(s) in stack	(internal, for Runtime only.)
3	application stopped between 2 progs	Runtime stopped between two steps during debugging
2	application stopped on SFC breakpt	(internal, for Runtime only.)
1	application stopped on error	Runtime stopped with a serious error, restart required.
0	application is running	TRUE = application is in "run" mode. FALSE = application pauses in "cycle to cycle" mode.

> **Info**
>
> *The driver variable SimulRTState with Offset 60 displays the status of Runtime on the server. This also applies is the variable is displayed on the standby server.*

Cyclical synchronization

Simulations run in a network on the server and standby server independently of one another. When the standby server starts, there is a synchronization with the server, however:

- ▶ This first synchronization is postponed by the packet runtime

- ▶ Small differences in the execution lead to growing diversion

The status of the simulated variables also therefore becomes different. In order to keep these differences low, use the `Transfer driver simulation image to standby` to synchronize the simulation image:

- ▶ create a new function

▶ Select **Transfer driver simulation image to standby**

▶ The dialog to select the driver opens

▶ In the list, all process-connected drivers (with the exception of the simulator) are available (clicking on the column heading changes the sorting)

▶ Select the driver that is to be synchronized

▶ Each time a function is called up on the standby server, an image is requested from the server for the selected driver

The differences in the current simulation status can be minimized through cyclical activation. Adapt the grid of the execution to the extent of the data to be synced, because this places a load on the network and computer.

The `Execution` of the function is set at `Standby` and fixed; it cannot be changed. It only happens if:

▶ The computer is the standby server

▶ The server is online

▶ The project is a network project

▶ Drivers were selected for syncing

TO CHANGE THE SELECTION OF THE DRIVERS

▶ In the driver properties, click in the `General` group in the `Parameter` properties

▶ The dialog for selecting the driver opens

zenon Logic Workbench

The `Simulation - programmed` is programmed in the in the zenon Logic Workbench, which works in close conjunction with the zenon Editor. For the simulation, (in contrast to the integrated solution) only the variables of a driver are exchanged between the editors.

DEFINITION OF THE VARIABLES TO BE EXCHANGED:

When starting zenon Logic Workbench, all driver variables that are not yet present in zenon Logic are created as global variables. The `Embed symbol` is automatically activated for these.

VISIBILITY

If a new variable is defined in Workbench in the `global` or `retain` area, this is only visible in zenon Logic. To make it visible in zenon too, the `Embed symbols` option must be activated. From this point in time, all variables are updated by both editors.

If the `Embed symbol` option is deactivated, the variable is deleted in the zenon editor. This action enables `global` or `retain` variables to be defined in zenon Logic, which are not visible for zenon.

For a newly-created variable created by the zenon Logic Workbench in zenon, an attempt is made to assign this to the **SPSMERKER** area. In the event that the zenon driver (on page 39) does not support the configured data type, a search is made for the first data area that supports the data type. If no area is found, the variable cannot be created.

A variable is only locked if in the case of a **multi-user project** and the `enable changes` status has not been set.

PROJECT SETTINGS

Some settings under *Project -> Project parameters... -> Further options-> Extended* in zenon Logic influence the functionality of the simulation.

Extended settings	Description
`Runtime: execute mode`	**Recommendation:** `Mode`: triggered `cycle time`: 1/4 to 1/3 of the driver update time for frequent value changes to simulated variables lead to increased load placed on the computer.
`compiler/options: Embed symbols of all variables`	**Recommendation:** Deactivate. `Active:` Data for a variable are exchanged, because the symbol names are assigned in Runtime. But: Changes to the configuration of the variables are not updated between zenon and zenon Logic. It is no longer possible to make changes online.
`compiler/options: Retain capitalization of symbols`	without effect: Variable names are always converted to capitals in Runtime.
`compiler/options: Create status bits for variables with a profile`	`Active:` The status of variables is simulated or reacted to in the simulation program.
`"C" compiler`	Native code is supported. Name of the DLL that the compiled code must contain: `t5app.dll.`

Runtime

zenon Logic Runtime offers the following functions:

- ► File transfer

- ► Data types:
  - BOOL
  - BYTE
  - DINT
  - DWORD
  - INT
  - LINT
  - LREAL
  - REAL
  - SINT
  - STRING
  - TIME
  - UDINT
  - UINT
  - USINT
  - WORD

- ► zenon arrays (see also Arrays and simulation (on page 53))

- ► Date stamp/time stamp

- ► Digital recording of values

- ► Dynamically linked function blocks, stored in `t5block*.dll`

- ► Hot Restart

- ► Logging:
  - Trace messages from the program `(printf)`
  - Error messages in Runtime
  - all sent to Workbench and to the diagnosis server

- ▶ Machine code: Native compiled code with C post compiler (not available under Windows CE)

- ▶ Online Change

- ▶ Programming languages: IL, AS, FDB, LD, ST

- ▶ Retain Data (remanant Data)

- ▶ Special function blocks and features:

  - Data serialization

  - dynamic memory allocation

  - embedded recipes

  - embedded variable lists

  - File Management

  - Files

  - Math

  - Random

  - Signals

  - String tables

  - Trigonometry

  - Time

- ▶ Spontaneous communication

  - Value change as an event

  - 16 connections

  - Hysteresis

- ▶ UDP & TCP/IP for IEC languages

- ▶ Variable interlocking

Arrays and simulation

To simulate multidimensional arrays at driver simulation, you must activate property 'Save complex variables in own segment' in the zenon Logic Workbench.

ARRAYS WITH START INDEX 1

If at a simulation arrays with start index 1 exist, these arrays are created with one more index in the zenon Logic Workbench.

> ### ▣ Example
>
> *In zenon an array has indices 1 to 4. This array has the indices 0 to 4 in the zenon Logic Workbench.*

This guarantees that the same indices are used in zenon and in zenon Logic. Index 0 is not transferred to the driver.

If you change array in the zenon Logic Workbench, you must consider this additional index.

Runtime files

When creating the Runtime files, a subfolder with the name of the simulation project is created in the `Runtime folder` of the project in the path `\RT\FILES\zenon\custom\drivers`. The file with the code of the simulation is `simulrt.cod` and is archived in this folder.

Once successfully compiled in the Workbench and transferred to the Runtime folder, the following message is shown in the output window of the Editor:

`SIMULRT.COD`

And if compiled with the C compiler, then also:

`T5APP.DLL`

REMOTE TRANSPORT

The Runtime files for the simulation are created as a subfolder of the driver files. For remote transport, the and all information contained in it is transferred to the target computer

Data exchange

The exchange of data from zenon Logic Runtime to the driver starts with the 2nd cycle of zenon Logic Runtime. Therefore the first run is available to initialize all variables in Runtime.

Data is only exchanged if:

- ▶ The variable is requested and:
  - did not yet receive a value
  - does not have the status `Switched off` (OFF/_VSB_N_UPD) or `Alternate value` (ALT_VAL/_VSB_AVALUE) in the driver
  - is still required by the driver.
  - has changed value or status

The exchange of data to the driver is carried out asynchronously to the Runtime cycle. The driver runs through the value changes once per update cycle. All values changed - with the exception of the `Switched off (OFF/_VSB_N_UPD)` or `Alternate value` (ALT_VAL/_VSB_AVALUE) status - are entered into the list of variables to be updated in the driver and sent to Runtime.

`TRANSFER DRIVER SIMULATION IMAGE TO STANDBY` **FUNCTION**

This function is only executed on the standby server (for configuration, see Cyclical comparison (on page 48) chapter). The data is synchronously fetched from the driver on the server that is compiling the image. The driver has 5 seconds of time for this. Because the data for the image is created in the same thread in which the simulation is running, it must be ensured that the simulation is processed within 5 seconds. If this time is exceeded, no image is transferred for this driver.

VALUE SIMULATION

Value changes are not transferred immediately, but stored in a buffer. The size of the buffer amounts to 8192 value changes or the five time the number of variables, according to whatever value is greater. The delay in transferring resulting from the driver cycle can be up to 100 ms. If the value changes again whilst it is waiting to be transferred, all values that have not yet been transferred are sent to Runtime and the new value is then noted. This can lead to an increased load for the computer and network, but ensures that all values are transferred in Runtime.

DRIVER WRITE COMMAND

If `simulation - programmed` is configured and active, the write commands at standby are passed through in Runtime.

Write operations are carried out by the driver asynchronously to Runtime. Data is exchanged via a buffer. The size of the buffer amounts to 8192 value changes or the five time the number of variables, according to whatever value is greater.

Write commands are ran through after their values are changed in Runtime. If a write command cannot be executed correctly, the error message **Write queue full! Write command for <DP-Name> lost!** is displayed.

If the status bits are activated, setting `Writing successful` will display if the value was written from the driver to the value of the variable. The Runtime program should delete this status again after processing the write command. A status bit that is already active does not cause any delay in the driver writing again. Write commands are also executed if the Runtime program was stopped by a breakpoint.

Note: Not all write commands from the driver must be visible in Runtime. If several write commands are in the buffer,only the last one is visible in Runtime.

SWITCH OFF VARIABLES, SWITCH TO SUBSTITUTE VALUE

If the variable is switched off or switched to a substitute value, this is not recognizable for Runtime. Changes in values that are also carried out by Runtime are not assigned to the variables in the driver. If the variables are again switched to spontaneous value or activated, the current value is again carried over from Runtime to the driver. When turning off the spontaneous value, status bit OFF is set. When you switch back to the spontaneous value, the OFF bit is reset.

Data storage

RETAIN DATA

Retain data is stored in the `SimulRt<TreiberID>.ret` file. One file such as this is created per simulation, provided retain variables are present. The file is taken into account when a comparison is carried out in the network.

DRIVER REMANENT

The `Simulation - programmed` setting at the driver start is not a criterion in order to restore the remanant variable image (on page 34).

Redundancy

START ON STANDBY SERVER

If the driver is already running on a standby basis, then the image for the simulation should already be available. In this case, the simulation is started "hot" with this image. If there is no image, a warm a start is carried out.

TRANSFER DRIVER SIMULATION IMAGE TO STANDBY FUNCTION

This function is only executed on the standby server (for configuration, see Cyclical comparison (on page 48) chapter). The data is synchronously fetched from the driver on the server that is compiling the image. The driver has 5 seconds of time for this. Because the data for the image is created in the same thread in which the simulation is running, it must be ensured that the simulation is processed within 5 seconds. If this time is exceeded, no image is transferred for this driver.

COMPARISON

In addition to the program files of the simulation, the following are also compared:

- ▶ Retain data (*.ret)

- ▶ all *.simul files
  This file extension is created by the simulation program for simulation specific files and archived
  in the folder with the computer description below the `Runtime folder`.

Status

STATUS SIMULATION

For the status simulation, in zenon Logic under *Project -> Project parameters -> Extended -> Compiler ->
Options -> Create status bits for variables with a profile* must be active. If the status of a variable is modified,
this triggers the transmission of a variable to the driver.

Changes to the following states does not trigger a value change at the driver and is also not taken over
from the simulation:

- ▶ `Real time external` (T_EXTERN/_VSB_RT_E)

- ▶ `Real time internal` (T_INTERN/_VSB_RT_I)

- ▶ `Standard time` (T_STD/_VSB_WINTER)

- ▶ `Writing acknowledged` (WR_ACK/_VSB_WR_ACK)

- ▶ `Writing successful` (WR_SUC/_VSB_WR_SUC)

- ▶ `Normal status` (NORM/_VSB_NORM)

- ▶ `Deviation from normal status` (N_NORM/_VSB_ABNORM)

- ▶ `Select in the network` (NET_SEL/_VSB_SELEC)

- ▶ `Runtime exceeded` (TIMEOUT/_VSB_RTE)

- ▶ `In process` (PROGRESS/_VSB_DIREC)

- ▶ `Switched off` (OFF/_VSB_N_UPD)

- ▶ `Substitute value` (ALT_VAL/_VSB_AVALUE)

> ### 💡 Info
>
> State bit `Writing successful (WR_SUC/_VSB_WR_SUC)` is available in the Runtime program and can be set back in order to display successful writing.
>
> However only setting back the state does not trigger the transfer to the <CD_PRODCUTNAME> Runtime. Only value changes are transferred from the driver to the Runtime.

STATUS BITS HANDLED DIFFERENTLY

The following status bits cancel each other out:

Bit	Description
`17: Spontaneous` `(SPONT/_VSB_SPONT)`	Status `SPONTAN` is set. This is the status set if status handling is turned off or no status was defined. Status `GI + INAVLID` is deleted.
`18: Invalid` `(INVALID/_VSB_I_BIT)`	Status `INVALID` is set. Status `SPONTAN + GI` is deleted.
`16: General` `interrogation` `(GI/_VSB_GR)`	Status `General Interrogation` is set. Status `SPONTAN + INAVLID` is deleted.
`8: Select in the` `network` `(NET_SEL/_VSB_SELEC)`	If this status bit is active (BSO activation/deactivation) the following status bits are accepted with `0` and also transmitted as `0` to the driver to identify a change: ▸ `Select in the network` (NET_SEL) ▸ `Cause of transmission` (COTx) ▸ `Select` (SE_870) ▸ `N_CONF` (P/N-BIT) ▸ `Test bit` (TEST) ▸ `Writing successful` (WR_SUC)

SELECT BEFORE OPERATE (SBO)

Status simulation must be available in the Runtime program in order for `Select Before Operate` to be reacted to. The procedure for `Select Before Operate` is defined in Runtime.

ACTIVATION

If an **SBO select** is sent to the driver, it triggers a value being written with state Select in the network (NET_SEL) + cause of transmission (COT_act).

A corresponding SBO procedure must be implemented and must start in Runtime.

The state bit Select in the network (NET_SEL) must be set back in the Runtime program.

DEACTIVATION

The deactivation triggers the writing of a values with status Select in the network (NET_SEL) + Cause of transmission (COT_deact).

The SBO procedure must be ended in Runtime accordingly.

The state bit Select in the network (NET_SEL) must be set back in the Runtime program.

Start / stop

DRIVER START

The value updating only starts at the standby server if the driver has received the process image from its server. The image must be received within the time out module, otherwise the driver will start without an image. If the standby server is upgraded to the server within the waiting waiting time, waiting is also ended. Drivers that start in a network project or without a network have no waiting time. This behavior applies for both the simulation as well as for the hardware mode.

RUNTIME START

If the driver starts the **simulation - programmed**, the first stage is loading the DLL with the simulation Runtime. After this, Runtime is parametered and the Runtime program starts, provided it can be successfully loaded.

If an image of Runtime is present, Runtime is started hot with this. Otherwise, a warm start is carried out. If this is also not possible, an attempt is made to start Runtime cold. If there is no valid program, Runtime boots up stopped. The status of all variables is set to INVALID.

The simulation program runs in it own thread and is therefore completely independent from the driver cycle.

> 💡 **Info**
>
> Retain variables
>
> *Retain data contain only the value of the zenon Logic variables not their status. This*
> *means for the start:*
>
> ▸ Warm start: The status which was set for a variable is restored - regardless of whether it is a retain
>   variable or not.
>
> ▸ Cold start with retain variables: Only the value of the retain variable in zenon Logic is restored, not
>   the status.

STOP RUNTIME

If Runtime is stopped, the status INVALID is set for all variables. Variables that are requested when Runtime is stopped have the status INVALID as initialization. Runtime secures the data for the warm start and online change:

   ▸ `SIMULRT.HOT`: contains the data for the hot restart.

   ▸ `SIMULRT.UPD`: contains the data for the online change.

Both these files are created in the Runtime program folder. Runtime must have write authorization for this folder.

RELOAD

   ▸ Recompiling the simulation project:
      causes the corresponding driver to be reloaded.

   ▸ Modification of the Time Out module:
      is not recognized as a change by Runtime and does not cause the driver to be reloaded.

Standard implementation in the driver kit triggers a hot restart when reloading the simulation project. If this is not possible (for example because the program is different), a cold start is carried out with `retain`.

REMOVING RUNTIME

If Runtime is running, this is stopped. Before Runtime is removed, all outstanding value updates are sent to the driver. Only then is Runtime released and `SimulRuntime.dll` removed from the PC.

Driver commands

Driver commands (on page 176) are used to influence drivers via zenon, e.g. start and stop. The engineering is implemented with the help of function `Driver commands`.

STOP/START DRIVER

If the driver is stopped in `simulation - programmed` mode, this will result in Runtime being removed from the memory. All variables obtain status `INVALID`. When the driver is restarted, the simulation is reloaded and Runtime is started.

SWITCH HARDWARE/SIMULATION DRIVER

When switching between modes `Hardware` and `Simulation`, the driver behaves according to the following pattern:

1. If the driver was configured for hardware mode in the Editor:

    - the driver is set to mode `simulation static` after function `Driver command` is executed with parameter `Driver in simulation mode`.

2. If the driver was configured in the Editor for one of the simulation modes (`static`, `counting`, `programmed`:

    - the driver is set to hardware mode and communicated with the control after function `Driver command` is executed with parameter `Driver in hardware mode`.

    - the driver then changes back to the configured simulation mode after function `Driver command` was executed with parameter `Driver in simulation mode`

Linking variables

The zenon driver is allocated to the zenon Logic variables via the names converted into capital letters. If the zenon Runtime requests a variable from the driver, it forwards this to the simulation. If no simulation is loaded, the variable receives status `Switched off` (`OFF/VSB_N_UPD`) when the driver is stopped. Otherwise it receives status `Invalid` (`INVALID/_VSB_I_BIT`). Variables of **driver variable** object type are never requested by the simulation.

time stamp

To use the time stamp, in zenon Logic under *Project -> Project parameters -> Extended -> Compiler -> Options ->Statusbit* create for variables with profile must be activated.

If in the Runtime program, the date for a variable is set, the value of the date and time is used as the time stamp. When the date or time changes during the last cycle, this always triggers a transfer of the value to the driver. This also applies if neither the value nor the status have changed.

Most drivers only transfer a new time stamp if the value and/or status change at the same time. An example of an exception is the IEC870 driver, which also transfers new timestamps from the hardware (or `Simulation - programmed`) with the same value and same status to Runtime.

> ### 💡 Info
>
> *It must be ensured that the date and time always increase. If this is not the case, this can lead to problems when archiving. The date must be a value greater or less than* 0.

Variables stamped by the Runtime program receive the status `Real time external` (T_EXTERN/_VSB_RT_E).

All variables to be transferred to the driver that are not stamped by Runtime receive a joint time stamp in the cycle. This ensures that the time corresponds to that of the change. These drivers receive the status `Real time internal` (T_INTERN/_VSB_RT_I).

DATE & TIME

The following applies for all date and time information:

- ▶ Times are UTC.

- ▶ All times must be between 02. 01. 1970 and 2038.

- ▶ Dates are converted into a string in the format YYYY/MM/DD.

- ▶ The time is converted into the format HH:MM:SS.

Please not for variables in simulation projects

The following is true for the creation and modification of variables in simulation projects:

1. zenon Logic

   If variables are created for a simulation project in the zenon Logic Workbench, they do not have a valid addressing for the communication. If the communication is switch to hardware, the communication can be interrupted when the new variables are used in screens without a adjusting the addressing correctly.

2. <u>Rename</u>

If a variable is renamed in a simulation project, the name of the variable is also changed in zenon accordingly. So for example you must adjust the names of variables which are used in a VBA project also in the VBA code.

3. <u>Integrated solutions</u>

Variables of integrated solutions such as zenon Logic, IEC870 or IEC850 must not be renamed in the simulation project. In this case the variables lose their project information. For example: If you rename `Project0/Global/NewVar` to `myNeVariablenwVar` in the simulation project, it becomes `myNewVar` in zenon. After the renaming a communication with the zenon Logic Runtime is impossible.

4. <u>Communication based on variable names</u>

Variables of a driver which communicates based on the variable names must not be renamed in the simulation project.

5. <u>Deleting or importing variables</u>

If a variable is deleted or imported while `online change` is activated in zenon Logic, all variables are removed and inserted again in zenon Logic.

6. <u>Variables with active "`embed`" symbol</u>

Variables which exist in a program and whoses "`embed`" symbol is active are not available via the Shared Memory interface in zenon. If a variable with an active "`embed`" symbol is moved from the global area to a program, the variable is deleted in zenon.

Error messages

Error messages in zenon editor:

Error message	cause and solution
The chosen name was already used for another driver or contains invalid characters! Please select another name.	The new name of the simulation project is already used in the project or is not valid. ▶ Give it a valid name
The simulation project from the **`<File name>/<Description>`** driver has not been compiled and cannot be supported! Please compile the project in Workbench.	A simulation project is available for the driver in Workbench. This was not compiled however. ▸ Start the zenon Logic Workbench with this driver's simulation project ▸ compile the project Tip: I does not make a difference if the diver settings are set to **`simulation - programmed`** in the driver settings
Write queue full! Write command for <Name Variable> lost!	Displays the loss of a write command.

2.5.6   Change driver

Drivers can be changed in zenon. The variables remain accessible and functional after the driver has been changed, provided that they are present in the driver that has been changed. The change of the driver definition is only possible for the released drivers which have been declared as being compatible.

> ⚠ **Attention**
>
> *Drivers that are linked to a zenon Logic project using the integrated solution cannot be exchanged.*

To change an existing driver:

1. Select the driver to be changed in the detail view

2. open the context menu with a right-click

3. Select the **`Change driver`** command

4. The same dialog dialog (on page 32) for creating a driver is opened

5. select the new driver

The necessary settings for the driver(s) (interfaces, interrupt address, etc.) are defined when the variables are configured.

> **Example**
>
> *The address concept for zenon is not available when a project is being created. Only the variables to be used have been defined.*
>
> *To allow parallel processing of a project, the `driver for simulator variables` is included as driver. The generation of the screens with dynamic linkages and the on-line test (setpoint input, etc.) can be carried out.*
>
> *Once the final address concept with the linkage to zenon is available, the driver is changed and can be used under project conditions.*
>
> *The change of a driver is only possible under the prerequisite that the driver used during the configuring and finally used in the end has the same properties (variable types).*

2.5.7 Delete Driver

To delete a driver, select the drive in the detail view in the context menu or use the `Delete` command in the toolbar. The deletion process must be confirmed with a confirmation request.

Attention: All variables belonging to it are also deleted.

2.5.8 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.

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.

> **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	12:00 AM	Connection in hold
LineStateConferenced	BOOL	12:00 AM	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	12:00 AM	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 `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	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 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	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

2.5.9   Driver documentations

Please find information on all zenon drivers in the corresponding driver documentation. You can find them in chapter Driver.

> ### 💡 Info
>
> *If you cannot find a driver documentation, you will receive help from support@copadata.com.*

2.6 Create, modify and use variables

Variables can be created:

▶ as simple variables (on page 74)

▶ 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. 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.h	Select the appropriate driver object type from the drop-down list.

`tm`)	
`Data type` (on page 12)	Select the desired data type (on page 12). Click on the ... button to open the selection dialog.
`Array settings`	Expanded settings for array variables. You can find details in the Arrays (on page 80) chapter.
`Addressing options`	Expanded settings for arrays (on page 80) and structure variables (on page 91). You can find details in the respective section.
`Automatic element activation`	Expanded settings for arrays (on page 80) and structure variables (on page 91). 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.

2.6.1   Simple variables

Simple variables always consist of a data type (on page 12) and a driver object type (cti.chm::/28685.htm). They form the basis for array variables (on page 80) and structure variables (on page 91).

Creating a simple variable

In the Project manager right-click on Variable and select New variable... in the context menu.

Define name, driver (C_Drivers.htm) , driver object type (on page 31) and data type in the following dialog.

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 the same name.
Driver	Select the desired driver from the drop-down list.
Driver object type	Select the appropriate driver object type from the drop-down list.
Data type	Select the desired data type (on page 12). Click on the ... button to open the selection dialog.

The newly created variable is displayed in the detail view of the project manager. You can edit additional properties of the variable in its property window.

Changing the properties of a simple variable

Changes of variable properties are done in the properties window. Select the variable to change in the detail view and enter the desired changes in the properties window.

Modify variable

The dialog `Modify variable` is opened if one ore more variables are selected and one of the following properties is selected:

1. Data type

2. Driver

3. Driver object type

4. Array settings

    a) Dim 1

    b) Dim 2

    c) Dim 3

    d) Start index

5. Offset calculation

6. Data type

Here, settings are changed for all variables which have been selected with multiple selection to make them uniform.

MODIFICATION, IF SEVERAL VARIABLES HAVE BEEN SELECTED:

If the value of the selected property is the same for all selected variables, it is displayed in the dialog; otherwise the entry stays empty. Empty means, that the setting has not been defined. If a selection has been done, it is no longer possible to make 'no selection'. If this is desired, the dialog has to be closed with `Cancel` and then reopened (exception: (Exception: checkboxes).

As long as the driver has not been specified, the combobox `Driver object type` lists all driver object types available for the driver of the selected variable. As soon as a driver is selected, only the driver object types of the specified driver are listed. If a driver from or to a zenon Logic driver is changed, the name of the variable is automatically changed and adopted.
With multiple variables selected, checkboxes can have one of three states: `on`, `off` and `undefined`. Fields that are not defined are not changed at the variable. If a field has been specified, it sets the new value for all selected variables.

CHECK BEFORE THE MODIFICATION:

For each variable before modification it is checked, if the resulting settings are allowed.

> **▣ Example**
>
> *Example for an invalid modification: An array variable and a simple variable have been selected; the array dimension is changed. Error: A simple variable must not subsequently be changed to an array variable.*

For variables that cannot be changed to the new settings, an error message is generated. Before the modification these are listed in the dialog '`Error modifying variable` (on page 78)', after the modification in the output window of the Editor.

Error modifying variable

If a modification for one or more variables is not possible, the dialog `Error modifying variable` opens. Here the variables that `cannot` be changed in this way, are listed. It is opened, `before` the modification is made and only if there are variables that cannot be changed. The entries can be sorted and filtered.

Parameter	Description
Name	Name of the variable, which cannot be changed.
Driver	Name of the driver with the driver identification, on which the variable is based.
Data type	Datatype used by the variable.
Driver object type	Driver object type that the variable uses.
Message	Description of the cause, why the variable cannot be changed.
Cancel	Modifying the variable is stopped. No modifications are made.
Ignore all	All variables, which cannot be changed, are ignored. The variables, which can be changed, are modified. A message for all ignored variables is generated in the output window. The text is the same as in the 'Message'.
Back	Return to modification dialog.
Status line	Number of selected variables and obstacles. The variable is considered non-modifiable once the first obstacle is recognized. Thee can be more obstacles, which then are not analyzed. In complex variables (arrays or structures) also all activated variables are checked, if they can handle the changes. A message is generated for each non-modifiable, activated variable. This means, that the number of obstacles can be higher than the number of selected variables.

Deleting simple variables

In the detail view, click on the variable to be deleted with the right mouse button to get the delete option in the following context menu:

Clicking on `Delete` opens the following security dialog:

2.6.2   Arrays

Arrays are fields of data types or variables. We make a distinction:

▶   Data type arrays: Can only be implemented in structures.

▶   Variables Arrays: Can also be implemented with simple data types.

Arrays can have up to three dimensions.

> **Example**
>
> *Example for a two-dimensional array variable:*
>
> *Engine speed [2.3]*
> *This variable consists of six variables (2x3):*
>
> ▸ Engine Speed [1.1]
>
> ▸ Engine Speed [1.2]
>
> ▸ Engine Speed [1.3]
>
> ▸ Engine Speed [2.1]
>
> ▸ Engine Speed [2.2]
>
> ▸ Engine speed [2.3]

Internal arrays are handled like structures (on page 91). That especially concerns addressing.

> ⚠ **Attention**
>
> *Array variables that are based on a stratonNG or straton32 driver must not start with* `1`*.*
> *A zenon array with dimensions* `1,2,2` *is created as a simple variable in zenon Logic.*

INACTIVE VARIABLES

Single variables of an array can be set inactive. Inactive variable

- ▸ Not available in zenon

- ▸ Are not registered with the driver

- ▸ Are not taken into account when the I/Os for the license size are calculated

In this way, reserve variables can be created in an array, which can be activated at a later point in time e.g. when expanding.

SIZE OF BLOCK ARRAY (UP TO VERSION 5.50)

Up to version 5.50, the size of the block array could be defined using the `Block array size` property. It is therefore possible to use only one licensed variable, for example, in the RGM or in VBA, but to indirectly address several variables with drivers that use offset addresses. This property remains for compatibility reasons.

Attention: The `internal driver` does not support this property. A value can be defined in the Editor, but no variables are displayed (value is always `0`).

Create array variable

Simple array variables are created like simple variables (on page 74).

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.
Driver	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	Select the appropriate driver object type from the drop-down list.

Data type	Select the desired data type (on page 12). Click on the ... button to open the selection dialog.
Automatic addressing	zenon calculates an automatic (on page 83) address allocation. This depends on the granularity of the driver object type used and the IEC data type on which the data type used is based.
Offset address follows data type offset and start offset	The offset follows the projected offset and bit for the structure element allocated.
Manual addressing	Manual (on page 87) addressing.
Each datatype starts with new offset	Defines how single array elements are positioned with automatic addressing. Active: a new offset is started for each array. Inactive: Offsets are used in full (variables are addressed in a row).
Activate all elements	On creating the new array variable all elements are automatically activated
Activate element manually	The elements are not activated when created. They have to activated in the variable list individually. This can be done via the context menu of the variable list.

Addressing

Arrays are usually addressed automatically. In doing so, the following is given for the array variables:

- ▸ Net address

- ▸ Start offset

- ▸ Data block (optional)

- ▸ Bit number (optional)

Nothing more has to be set for single array elements. The address is calculated automatically.

If arrays are to be addressed manually, the Manual addressing option has to be deactivated, either when creating the array or at a later point in time in the properties window.

The address is calculated again when changes are made to:

- ▶ Data type of an array

- ▶ Driver object type

- ▶ Driver

- ▶ Array dimension

Example for an array with automatic addressing for a INT variable

We would like to create an array of five integer variables.

BASIC CONFIGURATION

In the variables configuration dialog

- ▶ Name: Free issue

- ▶ Driver: S7 TCP/IP

- ▶ Driver object type: Ext. Data blocks

- ▶ Data type: INT

- ▶ Array: Start index is 1

- ▶ Dim 1: 5

- ▶ Dim 2: 0

- ▶ Addressing: Automatic addressing

- ▶ Each data type starts with new offset: active

- ▶ Element activation: Activate all elements

After clicking OK, the five defined variables are created in the variable list. Additionally a corresponding array variable is created in which we can set the array properties.

Status	Name	Unit	Net address	Data block	Offset	Bit num...	Alignm...
=..	A*	Filter..	Filter text	Filter text	Filter text	Filter...	Filter...
-	Array of INT		0	0	0	0	0
	Array of INT[1]		0	0	0	0	0
	Array of INT[2]		0	0	2	0	0
	Array of INT[3]		0	0	4	0	0
	Array of INT[4]		0	0	6	0	0
	Array of INT[5]		0	0	8	0	0

DETAIL CONFIGURATION

When selecting the array variable in the detail view, the properties of the whole array can be edited in the properties window. The most important settings are in the group "Addressing". There we set the following for the array:

- ▶ Data block

- ▶ Network address (=bus address)

- ▶ Start offset

As "Automatic addressing" was activated, all the addresses of the single array elements depend on this start offset. In our example, we change the `Data block` to `50`, we leave the `network address` at `0` and we change the `offset` to `100`.

The granularity of the S7 datablocks is 8 bit. So the datablock area is byte-oriented. Each INT variable needs 16 bits. This results in the following automatic addressing for our array:

Array	Net address	Data block	Offset
Array of INT[1]	0	50	100
Array of INT[2]	0	50	102
Array of INT[3]	0	50	104
Array of INT[4]	0	50	106
Array of INT[5]	0	50	108

Each offset is two higher than the previous, as a 16 bit variable needs two 8 bit offsets.

Taking a S5 driver instead of the S7 driver changes the addressing. On the S5, the datablock area has a granularity of 16 bits. The datablock area of a S5 PLC is word-oriented. Identical projecting would lead to the following result:

Array	Net address	Data block	Offset
Array of INT[1]	0	50	100NT
Array of INT[2]	0	50	101
Array of INT[3]	0	50	102
Array of INT[4]	0	50	103
Array of INT[5]	0	50	104

each offset now is only 1 higher than the previous, as a 16 bit variable needs only one 16 bit offset.

Example for an array with automatic addressing for a BOOL variable

We proceed as in the example of automatic addressing (on page 84) with INT data type; however we use the standard data type BOOL as a data type. All other settings stay as described there and we now create a new array variable.

We set the following values in the array variable properties:

```
Data block: 50

Net address: 0

Offset: 100

Bit number: 0
```

With an S7 driver, the following screen results with automatic addressing:

Array	Net address	Data block	Offset	Bit number
Array of INT[1]	0	50	100	0
Array of INT[2]	0	50	101	0
Array of INT[3]	0	50	102	0
Array of INT[4]	0	50	103	0
Array of INT[5]	0	50	104	0

A 1 bit variable has enough room in a 8 bit offset. Due to the `Each data type starts with new offset` option being activated, each array element starts a new offset.

If we deactivate this property in the array variable, we get the following addressing:

Array	Net address	Data block	Offset	Bit number
Array of INT[1]	0	50	100	0
Array of INT[2]	0	50	100	1
Array of INT[3]	0	50	100	2
Array of INT[4]	0	50	100	3
Array of INT[5]	0	50	100	4

Now the bits are all in one offset. If the offset would not be big enough (e.g. array of 20 BOOL variable), the next bit would be the first of the next offset. An array of 20 BOOL variable would be from offset 100 bit number 0 to offset 102 bit number 3.

Example for an array with manual addressing

For manual addressing, the `Manual addressing` property must be selected in the variable configuration. The the address information can be issued manually for each array element for:

- ▶ Net address

- ▶ Offset

- ▶ Data block (optional)

- ▶ Bit number (optional)

This could then look as follows for the example of array with automatic addressing for a BOOL variable (on page 86):

Array	Net address	Data block	Offset	Bit number
Array of INT[1]	0	50	100	0
Array of INT[2]	0	50	100	2
Array of INT[3]	0	50	101	0
Array of INT[4]	0	50	101	2
Array of INT[5]	0	50	102	0

⚠ Attention

If either the start offset of the array or the size of the array dimension is changed, all address information (offset and bit address) is newly calculated.

If the start offset is changed, the offsets of the variables already activated are also changed by the delta of the change. The offset difference between the activated variable and the array variables is thus retained.

Changing the properties of an array

The properties are changed using the Properties window and the `Modify variable` dialog.

Note two special properties in particular:

▶   `Last used offset` shows which offset (including bit number) was the last used in automatic addressing.

▶   `Size in bytes:` shows how many bytes are used by the selected array.

The following cannot be changed:

▶   Name

▶   Activating the elements

Property	Description
Driver	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	Select the appropriate driver object type from the drop-down list.

Data type	Select the desired data type (on page 12). Click on the ... button to open the selection dialog.
Automatic addressing	zenon calculates an automatic (on page 97) address allocation. This depends on the granularity of the driver object type used and the IEC data type on which the data type used is based.
Offset address follows data type offset and start offset	The offset follows the projected offset and bit for the structure element allocated.
Manual addressing	Manual (on page 101) or semi-automatic addressing.
Each datatype starts with new offset	Defines how single array elements are positioned with automatic addressing. Active: a new offset is started for each array. Inactive: Offsets are used in full (variables are addressed in a row).

NOTES ON CHANGING THE IEC DATA TYPE

If the data type of an array is changed, then the following applies:

▶ As long as the value of the current signal range is within the signal range of the data type, no amendment is made.

▶ The measuring range is not adapted.

Exception: Signal and measurements ranges are always adapted for the BOOL data type.

For the adaptation of the signal range, the new signal range must be less than the previous one:

▶ SingalbereichMin is only adapted if SignalbereichMinNeu <is SignalbereichMinAlt

▶ SignalbereichMax is only adapted if SignalbereichMaxNeu < is SignalbereichMinAlt

De/activating array elements

Each array element can be de/activated individually. Activating is either done on creating the array (activate all elements) or manually via the context menus of the single array elements. Inactive array

elements are handled by automatic addressing as though they were active. The inactive elements are place holders, that can be activated at any time.

2.6.3   Structure variables

A structure consists of individual elements. These are always data types. On creating a structure element always an existing data type has to be selected. Structure elements can be either embedded or linked.

Parameters	Description
`Embedded structure elements`	can have properties that differ from their basic datatypes. The properties are defined individually for each structure element and are saved there.
`Linked structure elements`	Always get all their properties from the basic datatypes that they are linked to. If the basic data type is changed, all linked elements are changed in the same way!

STRUCTURE DATA TYPES

You can find details on creating structure data types in the Data types (on page 12)/Structure data types (on page 22) section.

STRUCTURE VARIABLES

Structure variables are always based on a structure data type. So as a matter of principle they get the structure of the basic datatype. When creating a structure variable, a number of options can be defined:

Parameter	Description
Arrays	It is possible to select whether the structure is created as a single structure variable or as an array. Up to three dimensions can be defined for a structure array.
Addressing	The structure variable can now be addressed manually or automatically. As a default addressing is defined automatically. The calculation of the addressing thereby depends on the granularity of the driver object type that is used and the structure elements that are used. With automatic addressing, each data type can be started with a new offset.
Activating elements with structure variables	It can be set, whether all elements of the structure should be activated. Activating means, that all elements are set active. Active variables can be used in zenon at once. Inactive variables are not available to zenon, are not counted for the licensed I/Os and are not registered in the driver. Therefore reserve areas can be created in a structure, which can be activated at a later point in time. All elements of a structure variable can be activated or deactivated via the context menu at any time. **Note:** Deactivated structure variables are also taken into account when addresses are calculated. If this is not desired, the addresses must either be issued manually or the corresponding structure element must be deleted from the data type.

BEHAVIOR DURING XML IMPORT

STRUCTURES

Structures which differ from existing ones can be imported in already existing structures. Variables based on this are automatically adapted.

▶   The structure elements are identified by their name.

▶   At already existing structure elements the type is adapted if necessary.

▶   Non-existing elements are added.

▶   Elements which do not exist in the structure data type are removed.
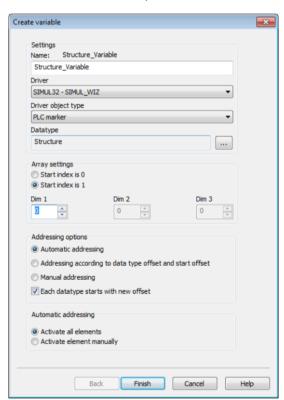
INACTIVE VARIABLES

At the import of structure variables, active and inactive variables are imported. Existing imports are not overwritten at the import. If an inactive variable is imported to a project and then activated, it stays active even after a new XML import.

Changing structure variables

To create a structure variable:

1. Create a simple variable (on page 72)

2. Select a structure data type as the data type

3. Select the driver and driver object type

4. Define the array dimensions

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.
Driver	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	Select the appropriate driver object type from the drop-down list.

Data type	Select the desired data type (on page 12). Click on the ... button to open the selection dialog.
Automatic addressing	zenon calculates an automatic (on page 97) address allocation. This depends on the granularity of the driver object type used and the IEC data type on which the data type used is based.
Offset address follows data type offset and start offset	The offset follows the projected offset and bit for the structure element allocated.
Manual addressing	Manual (on page 101) or semi-automatic addressing.
Each datatype starts with new offset	Defines how single structure elements are positioned with automatic addressing. Active: a new offset is started for each structure element. A bit offset > 0 increases the offset by 1. Inactive: Offsets are used in full (variables are addressed in a row).
Activate all elements	All elements are automatically activated when the new structure variable is created.
Activate element manually	The elements are not activated when created. They have to activated in the variable list individually. This can be done via the context menu of the variable list.

DATATYPE ORGANIZATION WITH AUTOMATIC ADDRESSING

▶ A driver BYTE granularity can only use the bit 0 to 7 with BOOL. All other data types start at bit 0.

▶ With WORD: BOOL bits 0 to 15, U(S)INT the bits 0 and 8. All other data types start at bit 0.

▶ With DWORD: BOOL bits 0 to 31, U(S)INT the bits 0, 8, 16, 24. (U)INT - the bits 0 and 16. All other data types start at bit 0.

▶ Strings must always start at bit 0.
As a result of this: If, with the current address, the > bit offset is 0, the variable is set to bit 0 and offset+1.

EXAMPLE

USINT = 8 bit, WORD granularity = 16 bit.

The variable can only start with the bits `0` or `8`. If, for example `2` is planned as a bit offset and the current address is `Offset=0 Bit=0`, the variable is set to `Offset 0 Bit 8`. If the current address were `Offset=0` and `Bit=7`, to `Offset=1 Bit=0`

Simple structure with automatic addressing

EXAMPLE

The first step is to create a structure data type. To do this, we create a `slider` structure with four elements:

- ▶ Engine speed set (UINT),

- ▶ Engine speed actual (USINT),

- ▶ Array [2] of
  - • on/off (BOOL) and

- ▶ monitoring (SINT).

	Structure Control	0	Structure datatype
	Speed Set	UINT	Structure element
	Speed Actual	UINT	Structure element
	On/Off[2]	BOOL	Structure element
	Monitoring	SINT	Structure element

Then we have to create a structure variable with the `Create variable` (on page 93) dialog, which is based on this data type. For this example, we use:

- ▶ the S7 TCP/IP driver

- ▶ the `Range` data block

- ▶ the array dimension `0`

- ▶ the activated options
  - • **Automatic addressing**
  - • **Each datatype starts with new offset**
  - • **Activate all elements**

The defined structure with all elements appears in the variable list after confirming the settings with **Finish.**

	SpeedControl	Structure Control
	SpeedControl.Speed Set	UINT
	SpeedControl.Speed Actual	UINT
	SpeedControl.On/Off[0]	BOOL
	SpeedControl.On/Off[1]	BOOL
	SpeedControl.Monitoring	SINT

When selecting the structure variable "Engine Speed Control" in the detail view, the properties of the whole structure can be edited in the properties window. The most important settings are in the group **Addressing.** There the `datablock`, the `net address` (= bus address) and the `start offset` can be changed. As `Automatic addressing` was activated, all the addresses of the single structure elements depend on this start offset.

In our example, we use the settings:

`Data block:` 50

`Offset:` 100

`Net address:` 0

The granularity of the S7 datablocks is 8 bit. So the datablock area is byte-oriented. Each UINT variable needs 16 bits, each USINT variable 8 bits, each BOOL variable 1 bit and each SINT variable 8 bits.

This results in the following automatic addressing for our structure:

Variable	Net address	Data block	Offset	Bit number
`Engine Speed Control.Engine Speed Set`	0	50	100	0
`Engine Speed Control.Engine Speed Actual`	0	50	102	0
`Engine Speed Control.Engine Speed.On/Off[1]`	0	50	104	0
`Engine Speed Control.Engine Speed.On/Off[2]`	0	50	105	1
`Engine Speed Control.Engine Speed.Monitoring`	0	50	106	0

"Last used offset": 106 bit 7

The individual elements of the structure variable are thus optimally fit into the offsets.

The 8 bit variable "`Engine Speed Control.Engine Speed.Set`" uses one offset, the 16 bit variable "`Engine Speed Control.Engine Speed.Actual`" uses two offsets, the two BOOL variable use two bit in one offset and the variable "`Engine Speed Control.Engine Speed.Monitoring`" uses one offset.

The addresses can be changed in the structure data type or in the address settings of the structure variable, but not in the single structure variable elements. That would only be possible with manual addressing.

Taking a S5 driver instead of the S7 driver changes the addressing. On the S5, the datablock area has a granularity of 16 bits. The datablock area of a S5 PLC is word-oriented.

Identical projecting would lead to the following result:

Variable	Net address	Data block	Offset	Bit number
`Engine Speed Control.Engine Speed Set`	0	50	100	0
`Engine Speed Control.Engine Speed Actual`	0	50	101	0
`Engine Speed Control.Engine Speed.On/Off[1]`	0	50	102	0
`Engine Speed Control.Engine Speed.On/Off[2]`	0	50	102	1
`Engine Speed Control.Engine Speed.Monitoring`	0	50	102	8

"Last used offset": 102 bit 15

The individual elements of the structure variable are again optimally adapted to the offsets here.

The 8 bit "`Engine Speed Control.Engine Speed.Set`" variable uses half an offset (bits 0 to 7), the 16 bit variable "`Engine Speed Control.Engine Speed.Actual`" uses one offset. The two BOOL variables only use the first two bits of the offset 102 and so the variable "`Engine Speed Control.Engine Speed.Monitoring`" starts in the same offset with bit number 8.

Structure variable as an array with automatic addressing

ARRAY WITH A STRUCTURE ELEMENT WITH SIMPLE DATA TYPE

Variables are addressed in packed form, based on the offset with the structure element, according to IEC data type and granularity. There is thus no difference to "packed" addressing.

STRUCTURE VARIABLE AS AN ARRAY

The address of the first structure element of the index `n` is derived from the address of the last structure element at the index `n-1` plus the offset at the structure element.

If the **`Each data type starts with new offset`** property is

 ▶   `active`: The index always starts at a whole offset (bit is `0`, maybe shifted by the offset at the structure element).

 ▶   `inactive`: is "packed" according to data type and granularity and shifted by the offset of the structure element. Regardless of this, the variable is set up according to data type and granularity.

EXAMPLE

We use the same example as in the Simple structure with automatic addressing (on page 95), however we change the array dimension `1` from `0` to `2` in the structure variable properties.

After clicking on **`Finish`**, the variable list looks as follows:

–	Speed-Control	Structure Control
	Speed-Control[0].Speed Set	UINT
	Speed-Control[0].Speed Ac...	UINT
	Speed-Control[0].On/Off[0]	BOOL
	Speed-Control[0].On/Off[1]	BOOL
	Speed-Control[0].Monitori...	SINT
	Speed-Control[1].Speed Set	UINT
	Speed-Control[1].Speed Ac...	UINT
	Speed-Control[1].On/Off[0]	BOOL
	Speed-Control[1].On/Off[1]	BOOL
	Speed-Control[1].Monitori...	SINT

The existing structure was changed to the effect that now each structure element is available twice.

For our example with the S7 driver, this results in the following automatic addressing, provided that we activated: `Each data type starts with new offset`:

Variable	Net address	Data block	Offset	Bit number
`Engine Speed Control[1].Engine Speed.Set`	0	50	100	0
`Engine Speed Control[1].Engine Speed.Actual`	0	50	101	0
`Engine Speed Control[1].Engine Speed.Ein/Aus[1]`	0	50	103	0
`Engine Speed Control[1].Engine Speed.Ein/Aus[2]`	0	50	103	1
`Engine Speed Control[1].Engine Speed.Monitoring`	0	50	104	0
`Engine Speed Control[2].Engine Speed.Set`	0	50	105	0
`Engine Speed Control[2].Engine Speed.Actual`	0	50	106	0
`Engine Speed Control[2].Engine Speed.Ein/Aus[2]`	0	50	108	0
`Engine Speed Control[2].Engine Speed.Ein/Aus[2]`	0	50	108	1
`Engine Speed Control[2].Engine Speed.Monitoring`	0	50	109	0

"Last used offset": 109 bit 7

The array structure was created exactly like a simple structure. The second structure array starts exactly, where the first ends.

Changing the driver to S5 results in the following:

Variable	Net address	Data block	Offset	Bit number
`Engine Speed Control[1].Engine Speed.Set`	0	50	100	0
`Engine Speed Control[1].Engine Speed.Actual`	0	50	101	0
`Engine Speed Control[1].Engine Speed.Ein/Aus[1]`	0	50	102	0
`Engine Speed Control[1].Engine Speed.Ein/Aus[2]`	0	50	102	1
`Engine Speed Control[1].Engine Speed.Monitoring`	0	50	102	8
`Engine Speed Control[2].Engine Speed.Set`	0	50	103	0
`Engine Speed Control[2].Engine Speed.Actual`	0	50	104	0
`Engine Speed Control[2].Engine Speed.Ein/Aus[2]`	0	50	105	0
`Engine Speed Control[2].Engine Speed.Ein/Aus[2]`	0	50	105	1
`Engine Speed Control[2].Engine Speed.Monitoring`	0	50	105	7

"Last used offset": 105 bit 15

EACH DATATYPE STARTS WITH NEW OFFSET

The array structure was created exactly like a simple structure. The second structure array starts exactly, where the first ends. Here, differences are possible, depending on the `Each data type starts with new offset` option: If this option is activated, the next structure array always starts at a new offset, even if the next structure would fit into the current offset.

> **Example**
>
> The last structure element is a SINT (8 bits) and uses offset 102 bits 0 to 7. For the example of the S5 driver (16 bit offsets) the next structure would start as shown below:
>
> ▸ Option "Each data type starts with new offset" activated: Offset 103 bit 0
>
> ▸ Option "Each data type starts with new offset" deactivated: Offset 102 bit 8. The structures are now directly after one another.

Manual addressing

Addresses can be issued semi-automatically or completely manually with manual addressing.

SEMI-AUTOMATIC SOLUTION

The offsets and the bit numbers are defined in the structure elements in the data type. But there only a relative offset is defined. In the structure variable, a start offset is defined and all relative offsets are added to this start offset.

EXAMPLE

As in Simple structure with automatic addressing (on page 95), we create the `structure slider` structure data type. A desired offset is then set for the individual structure elements, for example offset `3` for the `Engine speed set` structure element.

For structure variables, the `automatic addressing` option is activated when it is created. A start offset of `100` is defined in the properties of the structure variable. zenon then calculates, at the address of the "`Drehzahl-Regelung.Drehzahl.Soll`" variable, the offset `103` (100 from the structure + 3 of the structure element).

If the relative offset is to be subsequently changed, the following steps have to be carried out:

▸ First the offset in the structure element in the data type has to be changed.

▸ Then the start offset must be changed for the structure variable (for example to 101).

▸ And then back to the initial offset (100 in our example).

Background: The new address calculation is only done, when the start offset in the structure variable is changed, and not if only the offset in a structure element is changed.

FULLY MANUAL SOLUTION

Basically the projecting is the same as for the semi-automatic solution with the exception, that in the structure variable the addresses are entered by hand for each element.

If the start-offset of a structure variable is changed, the offsets of the variables already activated are also changed by the delta of the change. The offset difference between the activated variable and the array variables is thus retained.

Changing structure variables

Changes are made via the Properties and Variable dialogs.

The following cannot be changed:

▶   Name

▶   Activating the elements

Property	Description
`Driver`	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`	Select the appropriate driver object type from the drop-down list.
`Data type`	Select the desired data type (on page 12). Click on the ... button to open the selection dialog.
`Automatic addressing`	zenon calculates an automatic (on page 97) address allocation. This depends on the granularity of the driver object type used and the IEC data type on which the data type used is based.
`Offset address follows data type offset and start offset`	The offset follows the projected offset and bit for the structure element allocated.
`Manual addressing`	Manual (on page 101) or semi-automatic addressing.
`Each datatype starts with new offset`	Defines how single structure elements are positioned with automatic addressing. `Active:` a new offset is started for each structure element. `Inactive:` Offsets are used in full (variables are addressed in a row).

Changing the properties

Changes are made in the properties window.

Note that changes in the structure variable can affect the single elements of the structure variable (e.g. address calculation when you change the start offset or the driver). The properties of each element of the structure variable can be changed individually. Changes can either be made in the single elements or in the structure elements of the basic structure data type. The changes then are inherited. (See the Inheritance concept (on page 106) chapter in relation to this)

De/activating structure elements

The element has to be selected and then the appropriate command from the context menu is executed. Inactive array elements are treated as though they were active by the automatic addressing. The inactive elements are place holders, that can be activated at any time.

Inactive variable

- ▶ Not available in zenon

- ▶ Are not registered with the driver

- ▶ Are not taken into account when the I/Os for the license size are calculated

IMPORTANT NOTES

Note when deactivating variables:

- ▶ Variable references in screen elements are lost in deactivation. These connections can only be restored manually.

- ▶ If a structure element is based on an IEC data type that is not permitted with the selected driver, then this element cannot be activated.

- ▶ If variables are deactivated and then again activated, the dynamic elements for these variables are not updated in the Editor. In order to update them, click on the corresponding variable in the property window of the element.

Changing the sequence

The sequence of the structure elements of a structure variable can only be changed in the corresponding structure data type. When changing the structure data type all structure variables based on it are also changed!

Deleting structure variables

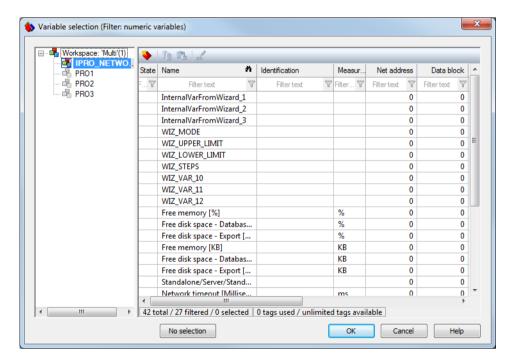Highlight the variable in the detail view and press the `Delete` key.

2.6.4 Project overlapping variables

Variables can also be selected from other projects and used throughout the project. This is also possible in different hierarchies. Variables from superordinate projects, projects at the same level and subordinate projects, but also variables from unconnected projects can be used as desired.

To use variables throughout a project, all projects concerned must, in Runtime:

- ▶ Be loaded

- ▶ Contactable in the network

    Hint: zenon does not check the contactability in the network in Runtime.

In the project, the variables that come from other projects are displayed with the following wording: `[Project name]#[Variable name]`, for example `Project1#WIZ_VAR_10`

If cross-project use is not possible, projects in the variable selection dialog are displayed as grayed out.

Note: If project names are subsequently changed or projects are copied, the correct assignment must be ensured. For this use the dialog for replacing project references.

⚠ **Attention**

Note the special conditions for cross-project variables in the command input.

2.7 Inheritance concept

`Inheritance` means, that the properties of an object are passed on to an object based on the first one. The inherited object in principle gets all properties from the basic object, but single properties or even all can be changed = overwritten.

In contrast there is also the prossibility of `linking`. Here again the new object gets all the properties from the basic object, but they `cannot` be changed/overwritten in the basic object. Changes are only possible in the basic object.

For both cases the following is true: If a property in the basic object is changed, all objects based on this basic object are also changed. So the single properties are not saved in the object, there is only a reference to the basic object.

Exception: As soon as a value is overwritten (changed), this property loses its reference to the basic object. If such an overwritten property is changed in the basic object, these changed no longer effect the inherited object. This is only true for the properties that lost their reference to the basic object. All other properties still have their reference to the basic object. An overwritten property can at any time be changed back to a referenced property.

2.7.1 Inheritance in zenon

In zenon the base objects are the data types. Both simple and structure data types link their properties to the variables based on them.

The variables are the objects based on the data types. In principle, they inherit all their properties. Exceptions are properties which

Can be only set at the variable, e.g. the addressing that is based on the driver object type and not the data type.

In the structures there are the structure elements. These can be either `embedded` or `linked`. If they are linked, they get all the properties from the data type that they are linked to. All changes in the basic data type are directly passed on. But no properties can be overwritten, as they only are linked and not inherited.

The embedded structure elements of a structure data type in principle do not have a reference to their basic data type at all. They are copies, where all properties can be changed individually. Changes in the basic data type do not effect them. Embedded structure elements can at any time be changed back to linked structure elements and vice versa.

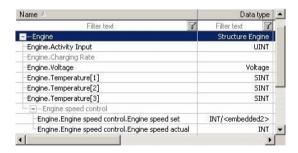**Inheriting properties with structure datatypes and structure variables**

Structure variables are handled in the same way as simple variables. But the properties have to be set for every single structure element. Each element of a structure variable is linked to a structure element of the structure data type it is based on.

Additionally a structure element again can be linked to a data type.

So the structure variable element inherits the properties from the structure element, which again can be linked to a data type.

It becomes even more complex, if structures are used in structures:

VARIABLE WITH STRUCTURE IN STRUCTURE

Name △	Data type
Filter text	Filter text
Engine	Structure Engine
Engine.Activity Input	UINT
Engine.Charging Rate	
Engine.Voltage	Voltage
Engine.Temperature[1]	SINT
Engine.Temperature[2]	SINT
Engine.Temperature[3]	SINT
Engine speed control	
Engine.Engine speed control.Engine speed set	INT/<embedded2>
Engine.Engine speed control.Engine speed actual	INT

In the example the variable is Motor.Drehzahl Regelung. Drezahl is deduced from the structure Motor. This again contains the structure element Speed Control, which again is a structure. This structure is linked to the Engine structure, as structures in structures can only be linked. In this structure, Speed Control is linked to the data type INT.

So if there are changes in this data type, these changes directly affect the variables Engine.Speed Control. Engine.Speed Actual.

So be careful with changes in data types.

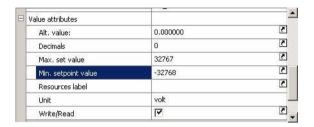Overwriting properties or restoring the inheritance from the data type are done as described above.

2.7.2 Inheriting properties of a datatype with simple variables

All properties are inherited from a data type when a new variable is created. This is marked with a checkmark in the property.

INHERITED PROPERTIES ARE MARKED WITH A CHECKMARK

In this example all properties are inherited from the data type except "Unit". This property was overwritten in the variable. If the unit of measurement property is changed in the basic data type, the unit of measurement property is changed for all variables based on this data type. But not in variable, where this property was overwritten (changed).

> ⚠ **Attention**
>
> *Please be aware, that changes to the data type always change all variables based on that data type (except properties that were overwritten accordingly).*

Overwriting properties

Properties of variables can be overwritten in two ways:

▸ The property is changed with the keyboard or the mouse As soon as the change is saved, the reference to the data type is lost.

▸ The reference of this property can cut from the data type via the context menu. In this case either only the selected property or all properties of the variable can be separated from the data type. (Screenshot!)

As soon as a property was overwritten, the checkmark disappears. All changes in the data type do no longer effect these overwritten properties of the variable.

Restoring the properties of a datatype

Select the overwritten property and in the context menu execute Link `all properties with datatype` or only `Link<Name>with datatype`. This property or all properties again are marked with the checkmark.

2.8　Value calculation

The value calculation defines the value range of a variable using different properties.

Fluctuating values can be picked up using different properties, in order to prevent unwanted reactions such as alarms or archive entries. In doing so, a distinction is made between:

- ▶ `Hysteresis`: Prevents the transfer of fluctuating values from the driver to variables.

- ▶ Hysteresis for archive: Prevents the creation of archive entries for fluctuating values in relation to the measuring range.

- ▶ `Threshold value`: Prevents unwanted reactions such as alarms for fluctuating numerical values (on page 114).

2.8.1　Hysteresis

Hysteresis defines the area within which a value change is ignored. This avoid fluctuating values that would constantly trigger limit breaches being displayed in a screen element.

In zenon, hysteresis can be stipulated in the properties of the variables (`Value calculation/Hysteresis`) for:

- ▶ Value calculations of the variables: `Positive for signal` properties and `Negative for signal` for signal range

  Values that change that are within hysteresis are not transferred to variables from the driver.

- ▶ Entries in the archive in the event of a value change: `Positive for archive` und `Negative for archive` properties for measuring range

  Changed values that are within the hysteresis are not entered into the archive.

> **▣ Example**
>
> *A limit value of 50° C is defined. The temperature normally fluctuates somewhat.*
> *Therefore the limit is breached each time the 50° limit is exceeded or gone under, which*
> *triggers an alarm and an entry in the archive.*
>
> *If hysteresis is defined for the variable:*
>
> ▸ `Positive for signal:2`
>
> ▸ `Negative for signal:2`
>
>   *And for the archive as hysteresis:*
>
> ▸ `Positive for archive:8`
>
> ▸ `Negative for signal:6`
>
>   *Then:*
>
> ▸ Only exceeding 52° and going below 48° is considered a limit breach
>
> ▸ Only once 58° is exceeded or 44° is gone below is an archive entry created

Note: Hysteresis that is to avoid unwanted reactions (such as alarms) to limit breaches due to fluctuating values can be created using threshold values (on page 114).

> **💡 Info**
>
> *Not every driver supports hysteresis. You can find information on whether your driver*
> *supports hysteresis in the documentation for the corresponding driver.*

2.9   Limits

Limits have the task to trigger a reaction, as soon as a limit or a status is violated or reached. The limit is entered in the Editor and fixed or is set variably depending on the value of a variable in the Runtime. The kinds of reaction can be manifold. For example: Color changes, entries in alarm and chronologic event list, function calls etc. are possible. The execution of the reaction is carried out in zenon.

The engineering of

> ▶ limits defines the behavior of `numerical variables`.

> ▶ `Status` defines the behavior for `binary variables` (low, high or logical 0, 1).

Note: For reasons of universal validity, the term "limit" also is used for "status" hereafter.

If an additional evaluation of the variable's status information is needed, this has to be carried out using a reaction matrix.

More information on reaction matrices can be found in the chapter Reaction matrix (on page 122).

USE WITH EPLAN

With an alarm, an **EPLAN** document can also be called up. To do this, the **EPLAN** connection must be licensed and the path to the **EPLAN** program must be defined in the **zenon6.ini** file. Configuration:

▸ Input of the character sequence **$PRG:xxxxxx** in the `Help file` property in the `Limits` group

> **Note: xxxxxx** is the placeholder for the parameters to be given

▸ The `Help chapter` property is ignored, but must not be empty

For details, see the Calling up EPLAN from the help chapter in the Runtime help manual.
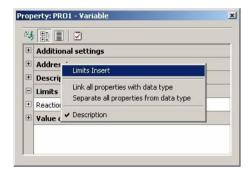
2.9.1   Defining limits in the Editor

Selecting a variable in the detail view of the Project Manager shows all the properties of the selected variable in the properties window.

To create a new limit, click `New limit`. Each time you click on `Insert limits`, a new limit (Limit 1, Limit 2, …) is added to the list.
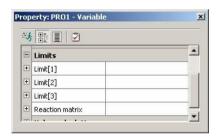
All available properties for defining limits are listed in the properties window and are described in the properties help. The context sensitive help is automatically displayed when selecting a property.

Any number of limits can be defined per variable.

Clicking on the plus in front of the property Limit opens the list. All defined limits are listed in this group. Now a limit can be selected to open the properties of this limit.

Delay

The `Delay time [s]` property can be used to configure a value for a time delay in the properties for the limit. This value defined the time range in seconds in which the limit breach must occur in order for the action defined for the limit breach to be carried out.

`Default value:` 0 - no delay

`Maximum value:` 4294967295 seconds

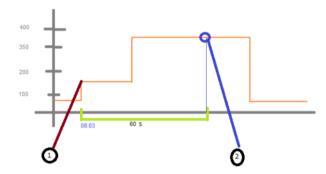The delay time applies regardless of which way the limit as breached.

Example:

- ▶ Engineering
  - **Limit value 1** has a value of 50.
  - **Limit value 2** has a value of 100.
  - The delay time is 3 s.

1. The value increases to `110`.

   • After the breach has occurred for 3 seconds, the limit breach for the value `100` is displayed.

2. The value decreases to `70`.

   • Once this value has lasted for `3` seconds, the alarm for the limit breach 100 is deactivated and a limit breach for the value `50` is displayed.

3. If the value goes below `40`, the alarm is deactivated after `3` seconds.

TIME STAMP AND VALUE

The time of the first violation is entered as the time stamp for the limit violation in the AML and the CEL. The first value after the delay time is entered as value, not the value that triggers the alarm.

Key:

Digit	Description
1	Time of limit breach. Start of the configured delay time. Value to be triggered.
2	End of the configured delay time. Alarm received: ▸ Time: Time of limit violation (1) ▸ Value displayed: Value after the delay time (2)

> ⚠️ **Attention**
>
> *In combination with the `Flashing` property, this property has the following effect in the network:*
>
> ▸ `Limit value with flashing and without delay:`
> Linked elements will flash on the server and on the clients.
>
> ▸ `Limit value with flashing and delay:`
> Linked elements will only flash on the server, not on the clients.
>
> ▸ `Alarm with flashing and without delay:`
> Linked elements will flash on the server and on the clients.
>
> ▸ `Alarm with flashing and delay:`
> Linked elements will flash on the server and on the clients.

Threshold

Thresholds prevent unwanted reactions to fluctuating numerical values. They create a limit hysteresis. The limit is therefore a different one for the start of the limit breach than the end of the limit breach.

For example: Limit max. at `100`, threshold value `10`. As soon as the value of the variable reaches `100`, the limit is violated. If the values goes down to `95`, nothing happens. Only once the value goes down to `89` does the limit violation end.

Thresholds can be set for maximum or minimum. For maximum, the threshold moves downwards, and upwards for minimum.
Example of maximum value: Maximum value = 900, threshold = 10, results in 890
The limit will be violated from a value of 900 and the critical area will be left when falling under 890.
Example of minimum value: Minimum value = 900, threshold = 10, results in 910
The limit will be violated from a value of 900 and the critical area will be left when it goes above 910.
In contrast to hysteresis (on page 109), this property does not influence the display of the value, only the alarm behavior.

Deduce limits from datatypes

Limits can already be defined in the datatype. The limits engineered there are available in `all` variables based on that data type. The inheritance concept works here as described above: all properties are inherited from the datatype and can be overwritten in each single variable. So a limit also can be

deactivated in a single variable. A limit inherited from the datatype `cannot be deleted` in a single variable (deleting is only possible in the datatype). But further limits can be added in the variable. These variable-specific limits of course can be deleted in the variable.

More information on reaction matrices can be found in the chapter Datatypes (on page 12).

Multiple selection

If several variables are selected, the properties of all these variables can be edited at the same time. If one of the selected variables does not have the limit to be edited, it will not be added there.

With this multi-selection properties, that differ from variable to variable, are displayed in red in the properties window. (The property of the first selected variable is displayed). If properties do not exist for all the selected variables (e.g. limits), they are marked in yellow.

Deleting limits

Click "delete limit" in the properties.

Overlapping limits

Analog limits can be defined overlapping. In this case zenon automatically checks which limit is valid.

> **Example**
>
> *Limit 1 (maximum) is defined 100. Limit 2 (maximum) is defined 200. If the variable in the Runtime gets a value between 100 and 199, limit 1 is violated. If the value is higher than 200, limit 2 is violated.*

2.9.2   Limits in the Runtime

For binary variables zenon reacts on reaching the status (positive edge). For numerical variables the control system reacts on reaching or violating the defined maximum (equal or bigger) and reaching or violating the defined minimum (equal or smaller). This is called a limit violation.

> **Info**
>
> *A binary variable has two limits, a Min limit and a Max limit. Care that: Enter the Min limit as limit 1, and the Max limit as limit 2.*

At the calculation of limits of numerical variables, the raw value of the variable (signal resolution) for the use in zenon is always calculated in a linear way unless it is defined differently with function 'non linear value adjustment".

Example: A temperature sensor sends its value to the control. The corresponding byte variable has a signal resolution of 0...255. In zenon the corresponding variable has a measuring range from 0..100. Hence:

Control	Control system
0	0
127	50
255	100

If a limit is violated, when zenon is started, the reaction is executed immediately on starting the Runtime.

Exception: If the reaction should be an entry in the alarm or CEL list, the system first checks whether the entry already exists from the previous session. If this is true, the entry is `not` done again! So double entries are avoided. All other reactions are nevertheless executed.

> ⚠ Attention
>
> *Only one limit of a variable can be active at a time.*

More information on the conversion of the signal range to the measuring range can be found in the properties help section of the properties window if you select the "Value" property.

End of a limit violation

If a limit is violated, a value change within the limit range does not lead to a new reaction.

For example: A limit range of `100-200` is configured. If the value changes from `110` to `115`, there is no further reaction. If the value changes to `210`, there is a reaction again because of the next limit.

Unwanted reactions for fluctuating numerical values can be avoided by using thresholds (on page 114).

2.9.3   Dynamic Limit Text

The dynamic limit text allows it to include the current values of other variables in the limit text of a variable or a Reaction matrices (on page 122) (Rema). At that a fixed text can be linked via an index with key words generated from a language table (see also Using key words in text lists (on page 120)). In the Runtime this limit text can be displayed in the Alarm Message List, Chronologic Event List and/or the alarm status bar. In the dynamic elements `Link text` and `Status` the dynamic limit text cannot be displayed.

SYNTAX

The following nomenclature is valid for the parameterization of the limit texts:

Parameters	Description
$	Denotes a dynamic limit text; this character must be on the first position in the limit text.
;	Separator of commands. Is used in order to separate the constant text from the dynamic text. When separating variables, no space must be left between the separator and the variable.
@Text	Text from the currently loaded language table
%PV	Value of another variable with the name „PV"
@Text+%PV	Composed keyword for the text from the currently loaded language table. Here the value of variable 'PV' is saved and when being displayed a text is searched in the language table which is a combination of 'text' and variable value.

This text encoding works only for the Alarm Message List or the Chronological Event List. In text elements this functionality is not available. The using of combined keywords makes it possible to create dynamic limit texts which can have more than 80 characters.

Example 1

Limit text = $@Text; X-Pos ;%Value1; Y-Pos ;%Value2

The following preconditions are valid:

- ▶ @Text is a key word in the language table corresponding to „Text from table"

- ▶ Value1 is an integer variable with the value 14

- ▶ Value2 is an integer variable with the value 12

Output = Text from table X-Pos 14 Y-Pos 12

Example 2

$OFF in position ;%String1

The following preconditions are valid:

- ▶ %String1 is a string variable with the content „ORT"

Output = OUT in position ORT

> **Example 3**
>
> *$%@message_+value1*
> *The following prerequisites apply:*
>
> ▶ `@message_` is a part of a keyword in the language table
>
> ▶ `Value1` is an integer variable with the value 12
>
> *With the combined keyword, the result is the keyword „message_12" and the corresponding text from the language table*

> **Example 4**
>
> *$%@message_+String1*
> *The following prerequisites apply:*
>
> ▶ `@message_` is a part of a keyword in the language table
>
> ▶ `String1` is a string variable with the content „zenon".
>
> *With the combined keyword, the result is the keyword „message_zenon" and the corresponding text from the language table*

LIMITS FOR DYNAMIC LIMIT TEXTS

The storage of dynamic limit texts leads to a number of limits:

1. Static text is stored in the storage area of the limit text:
   maximal 128 characters

2. Values of the linked variables are stored in the storage area of the commentary text:
   Maximum 80 characters.

3. If dynamic limit texts are used, no commentaries can be used for these alarms.

This limits can be evaded:

EXTENSION FOR LONG DYNAMIC LIMIT TEXTS.

Variable values for dynamic limit texts can be evacuated in an own file.

Requirements: In the project properties in group `Alarm Message List` the property `Long dynamic limit texts AML` or in group Chronological Event List the property `Long dynamic limit texts CEL` is active.
Instead of the commentary field the values are stored in an own file. 264 bytes are available per limit. Per variable 10 bytes of these are used for internal structure information.

Example 1: Dynamic limit with a string variable:
Of the 264 byte

- ▶   10 byte are used for structural information

- ▶   254 byte are available for the content of the string variable

Example 2: Dynamic limit with a string variable and a word variable:
Of the 264 byte

- ▶   20 byte are used for structural information

- ▶   2 byte for the content of the word variable

- ▶   244 byte are available for the content of the string variable

The strict limitation to three variables per limit text no longer exists.
The file names for the additional created files are `Dxxx.aml` or `Dxxx.cel`.
For the ring buffer the additional information is saved in `aml.bin` und `cel.bin`.

ERROR TEXTS DURING RUNTIME:

Parameters	Description
XXX	The value of a linked variable has changed.
---	values of linked variables are not available (can happen at program start).
>3	number of linked variables is higher than 3 (so value cannot be saved).

Dynamic key words in limit texts

Alarm messages from Boolean variables do not provide the reason why an alarm was triggered when dealing with collective messages. The cause can be transferred using an index variable, so that the limit

text prints out the cause for the alarm anyway. The correct text is found and displayed using the dynamic limit text (see Language switching chapter). For this, no fixed word can be used as a key word, but instead a dynamic one. This is comprised of **prefix+value of index variable**:
`%@errornr+mIndex`

It is possible to use multi-dimensional key words (several indices).
For example: `fixed text; @Motorfehler%meinIndex1-%meinIndex2` becomes, for example `@Motorfehler4-67` in Runtime. This key word is then searched for in the language table and the respective text is displayed.

EXAMPLES

EXAMPLE 1

`$@error in module: ;%moduleNr;@error type: ;%@errornr+mIndex;`

Description:

`$` defines a dynamic limit value text.

`@error in module`: a normal key word from the language table

`%moduleNr`: Is replaced by the value of the **ModuleNr** value in Runtime (Functionality of the dynamic limit text)

`@error type`: a normal key word from the language table

`@errornr`: The first part of the dynamic key word

`+mIndex`: Is replaced in Runtime by the value of the **mIndex** variable and supplements the keyword **@errornr**

Result:

Value of `mIndex`	Keyword
1	**@errornr1**
2	**@errornr2**

EXAMPLE 2: STRINGS

`$%@Message_+Internal_UINT_001;/;%@Reason_+Internal_STRING_004`

If `Index_UINT_001` has the value `4` and `Index_STRING_004` has the value `'Fire'`, then the texts **Message_4** and **Cause_of_fire** are searched for in the language table and displayed in the AML/CEL.

2.10 Reaction matrices

> 💡 **Info**
>
> *Limit information can be defined centrally in a reaction matrix (abbr. rema).*
> *When for example for a number of filling levels the same filling level control is carried*
> *out, it can be defined in a single reaction matrix. This makes it possible to change for*
> *example the value of the lower limit at a central point.*

ADVANTAGES WITH RESPECT TO LIMITS:

▶ Reaction matrices can be used multiple times. A reaction matrix can be linked to several variables that even can be based on different data types.

▶ Reaction matrices can handle each value change of a numeric variable as a new limit violation. This is not possible in limits.

▶ With reaction matrixes, a status such as invalid, for example, (see status processing) can be requested. This is also not possible with limit values.

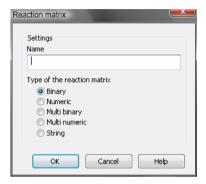More information on limits can be found in the chapter Limits (on page 111).

2.10.1 Creating a reaction matrix

Select "Reaction matrix" in the Project Manager and open the context menu with the right mouse button. There, select **Reaction matrix new....**

In the appearing dialog, enter the name and the type of the reaction matrix. Please be aware that the type cannot be changed later on. The name however can be changed at any time.

The several types of reaction matrices are used for the following evaluations:

Parameters	Description
Binary	Evaluation of discrete states (bit-oriented)
Numerical	Evaluation of analog states (value-oriented)
Multi binary	Extended evaluation of 32 bit variables; the first 16 bits determine the value of the variable and are passed on as numeric value; rerouting of status bits; special function (switch-off of data points when status bit is set) (Protective data configuring - only SAT-specific)
Multi numeric	Extended evaluation with simultaneous status routing and special function (switch-off of data points when status bit is set)
String	collation

2.10.2   Editing a reaction matrix

In order to edit a reaction matrix:

▶   select the reaction matrix from the detail view

▶   select the desired command from the context menu
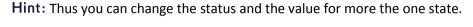
CONTEXT MENU DETAIL VIEW REACTION MATRIX

Parameter	Description
New reaction matrix	Creates a new reaction matrix.
Configuration	Opens the dialog for configuring the reaction matrix.
Copy	Copy selected reaction matrix.
Paste	Paste copied or cut reaction matrix.
Delete	Delete selected reaction matrix.
Export selected XML	Exports selected reaction matrices as an XML file.
All filters removed.	Removes all filter settings.
Edit selected cell	Opens cell for editing if the cell is allowed to be edited in the list.
Properties	Opens the property window.
Help	Opens the online help.

DIALOG STATES AND REACTIONS

By clicking Configuration the dialog for configuring the states is opened. The configuration possibilities depend on the type of reaction matrix.

In list States you can also select more than one status at a time and edit them. Different settings are highlighted with the help of a red background or a red frame.

Hint: Thus you can change the status and the value for more the one state.

CONTEXT MENU STATES

In the list field States the following commands are available in the context menu:

Command	Description
Creating a new status	Creates a new status based on the currently selected state.
Create a new sub-status	Only for multi-binary: Creates a new sub-status for the selected status.
Copy	Copies selected status to the clipboard. **Note for multi binary**: You can only copy main states. Sub-stated are copied together with their main states.
Paste	Pastes the selected status from the clipboard.
Delete	Deletes selected status from the list. The default status cannot be deleted.
Test	Opens the dialog for testing the status.
Up	Moves selected entries up. (Also possible with Drag&Drop.)
Down	Moves selected entry down. (Also possible with Drag & Drop.)
Help	Opens the online help for the respective reaction matrix.

You can find general settings and the definition of the values at the respective Rema:

▶ Binary (on page 127)

▶ Numeric (on page 132)

▶ String (on page 136)

▶ Multi-binary states and reactions (on page 143)

▶ Multi-numeric states and reactions (on page 146)

> **Info**
>
> *In case of reload or Server-Standby Switch, the present responses or writing affirmations are distorted.*

2.10.3   Types of reaction matrices

There are five types of reaction matrices available:

For the types multi binary and multi numeric you can also configure AIL/CEL text (on page 151), Status routing (on page 151) and variable treatment (on page 152).

Binary

A binary reaction matrix is used for the evaluation of any kind of flags and status bits of variables. The evaluation is bit-oriented. Also analog variables can be evaluated in this way.

Parameter	Description
States	List of the engineered states with value, state and cause of transmission.
New status	Creates a new status based on the currently selected state.
Sub-status	Only for multi-binary: Creates a new sub-status for the selected status.
Delete	Deletes all selected statues from the list.
Test	Opens the dialog (on page 154) for testing the status.
Up	Moves selected states up.
Down	Moves selected states down.
Status	Selection of the states.
Cause of Transmission	Additional status for cause of transmission.
Function	Opens the dialog for linking a function. You can deselect an already linked function with the help of button **No selection** in this dialog.
Call via the Alarm Message List	Only available if you activated option In Alarm Message List in **AML/CEL**. Active: Function is called via the AML.
Additional attributes	
Limit color	Color when a limit has been violated.
Flashing	Active: Flashes when a limit has been violated.
Invisible	Active: Will be switch to invisible when a limit has been violated.
Help file	Clicking on ... opens the dialog to open a help file in **chm** format. It must have already been created in the project manager under **files/help**. In Runtime, the linked help file is opened if the corresponding alarm is selected and the **Call help** button is clicked. **Note:** The property can only be configured if the In Alarm Message List property is active.

`Help chapter`	Indication of the help chapter. **Note:** Only available if the `In Alarm Message List` property is activated.
`Additional information 1`	In the Runtime the additional information entered can be assessed in a VBA macro.
`Additional information 2`	In the Runtime the additional information entered can be assessed in a VBA macro.
AML/CEL	
`In Alarm Message List`	`Active`: Will be entered in the AML.
`To acknowledge`	Only available if `In Alarm Message List` is activated. `Active`: Must be acknowledged.
`Comment required`	To be able to acknowledge the alarm, a comment must be entered beforehand. The user must be authorized to carry out the necessary function.
`To delete`	Only available if `In Alarm Message List` is activated. `Active`: Must be deleted.
`Print`	Only available if `In Alarm Message List` is activated. `Active`: Will be printed via the set standard printer.
`In Chronological Event List`	`Active`: Will be entered in the CEL. **Hint**: If the initial value (the first value that comes from the controller) or the value when Runtime is started already violates the limit value or the Rema status is active as a result, no entry is created in the CEL. Only once the limit violation has been rectified and then is violated again, or the state becomes inactive and then active again, is a CEL entry generated.
`Alarm/event group`	Allocation to an alarm/event group.
`Alarm/event class`	Allocation to an alarm/event class.

VALUE

Parameter	Description
`Value`	Current value of variable.
`Limit text`	Text which is displayed when at a limit violation.
`Delay`	Time period which the limit violation must last in order for the limit to be active.
`Status text for counter in the mathematics driver`	Assignment of four possible status numbers for a counter (mathdr32.chm::/23818.htm) in the mathematics driver. For this the variable must have a reaction matrix.

The definition of the combination of state and/or status value is made by way of releasing the option fields. The state value is the binary encoded `non-scaled value` of the variables. The value transformed into the measuring range (linear or non-linear value adjustment) is not considered!

The bit assignment is:

Status:	
`Non-scaled value bit "0" set`	2^0 = 1
`Non-scaled value bit "1" set`	2^1 = 2
`Non-scaled value bit "2" set`	2^2 = 4 ....

Two linkages are possible. The meaning of the fields is

Status:	
`not considered`	dot, not set
`check for "0"`	0, bit is checked for logical value "0"
`check for "1"`	1, bit is checked for logical value "1"

The sequence of the configuring is decisive for the later use. During the processing of the reaction matrix in online operation the check of the state and status bit combination is started in the top line. The check in the Runtime system is continued until the line which meets the given states. All other states are not checked further. If no line fulfills the current variable condition, the default entry applies. This enables prioritization of the limit texts for the chronological event list, alarming and the alarm information list. The status evaluation is done in the same way as the value evaluation of the variable.

ANALYSIS

In binary variables only bit 0 is evaluated. The other bits are ignored, In 8 bit variables (IEC data types SINT, USINT) only the bits 0 to 7 are evaluated, and so on.

Example	(single information and monitoring for disturbance)
check whether INVALID-bit is set	……………., …………1…
Check whether bit 0 is set	……………0, …………….
Check whether bit 1 is set	……………1, …………….

In the above example, irrespective of the condition of the single information, the check is completed after the first line when the INVALID bit is present (no connection to PLC - data point disturbed). The variable properties (alarm, blinking, CEL, etc.) are controlled exclusively by this condition. Once the INVALID bit is no longer present, the other lines are processed. If the sequence is to be changed, then this can be effected with the selection of the respective line and the operations **New, Delete, Upwards** und **Down.**

Numeric

A `numeric reaction matrix` is used for the evaluation of any kind of limit states and status bits of the variables. In contrast to the binary matrix here the numerical values are evaluated. Not the PLC value but the adjusted measuring range value (linear or non-linear value adjustment) is used.

Parameter	Description
`States`	List of the engineered states with value, state and cause of transmission.
`New status`	Creates a new status based on the currently selected state.
`Sub-status`	Only for multi-binary: Creates a new sub-status for the selected status.
`Delete`	Deletes all selected statues from the list.
`Test`	Opens the dialog (on page 154) for testing the status.
`Up`	Moves selected states up.
`Down`	Moves selected states down.
`Status`	Selection of the states.
`Cause of Transmission`	Additional status for cause of transmission.
Function	Opens the dialog for linking a function. You can deselect an already linked function with the help of button `No selection` in this dialog.
Call via the Alarm Message List	Only available if you activated option In Alarm Message List in `AML/CEL`. Active: Function is called via the AML.
`Additional attributes`	
Limit color	Color when a limit has been violated.
Flashing	Active: Flashes when a limit has been violated.
Invisible	Active: Will be switch to invisible when a limit has been violated.
Help file	Clicking on `...` opens the dialog to open a help file in `chm` format. It must have already been created in the project manager under `files/help`. In Runtime, the linked help file is opened if the corresponding alarm is selected and the `Call help` button is clicked. **Note:** The property can only be configured if the In Alarm Message List property is active.

Help chapter	Indication of the help chapter. **Note:** Only available if the `In Alarm Message List` property is activated.
Additional information 1	In the Runtime the additional information entered can be assessed in a VBA macro.
Additional information 2	In the Runtime the additional information entered can be assessed in a VBA macro.
AML/CEL	
In Alarm Message List	`Active`: Will be entered in the AML.
To acknowledge	Only available if `In Alarm Message List` is activated. `Active`: Must be acknowledged.
Comment required	To be able to acknowledge the alarm, a comment must be entered beforehand. The user must be authorized to carry out the necessary function.
To delete	Only available if `In Alarm Message List` is activated. `Active`: Must be deleted.
Print	Only available if `In Alarm Message List` is activated. `Active`: Will be printed via the set standard printer.
In Chronological Event List	`Active`: Will be entered in the CEL. **Hint**: If the initial value (the first value that comes from the controller) or the value when Runtime is started already violates the limit value or the Rema status is active as a result, no entry is created in the CEL. Only once the limit violation has been rectified and then is violated again, or the state becomes inactive and then active again, is a CEL entry generated.
Alarm/event group	Allocation to an alarm/event group.
Alarm/event class	Allocation to an alarm/event class.

VALUE

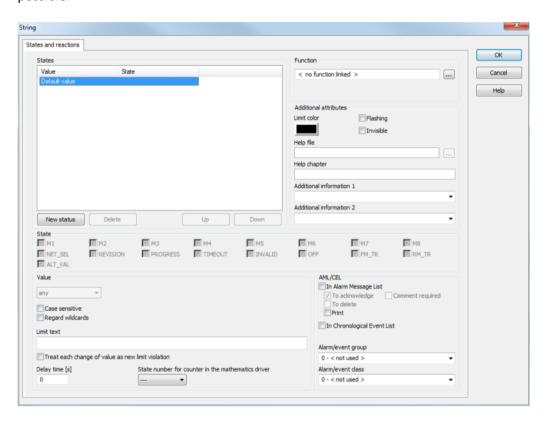| Parameter	Description
Value	Definition of the declaration of value: ▶ any: any change of value violates the limit ▶ greater: Enter of a limit ▶ smaller: Enter of a limit ▶ equal: Enter of a limit ▶ Area: Enter of an area (from ... to)
Limit text	Text which is displayed when at a limit violation.
Treat each change of value as new limit violation	Active: Each change in value is displayed as a limit violation.
Delay	Time period which the limit violation must last in order for the limit to be active.
Status text for counter in the mathematics driver	Assignment of four possible status numbers for a counter (mathdr32.chm::/23818.htm) in the mathematics driver. For this the variable must have a reaction matrix.

💡 Info

A decimal value can be entered with either a `comma` *or a* `point` *as a decimal separator, it will automatically be changed to a* `point`.

String

A string reaction matrix is used for the evaluation of limit values and status bits of the string variables. In addition to the known configurations of limit value and status combinations, further configurations are possible.

Parameter	Description
States	List of the engineered states with value, state and cause of transmission.
New status	Creates a new status based on the currently selected state.
Sub-status	Only for multi-binary: Creates a new sub-status for the selected status.
Delete	Deletes all selected statues from the list.
Test	Opens the dialog (on page 154) for testing the status.
Up	Moves selected states up.
Down	Moves selected states down.
Status	Selection of the states.
Cause of Transmission	Additional status for cause of transmission.
Function	Opens the dialog for linking a function. You can deselect an already linked function with the help of button **No selection** in this dialog.
Call via the Alarm Message List	Only available if you activated option In Alarm Message List in **AML/CEL**. Active: Function is called via the AML.
Additional attributes	
Limit color	Color when a limit has been violated.
Flashing	Active: Flashes when a limit has been violated.
Invisible	Active: Will be switch to invisible when a limit has been violated.
Help file	Clicking on ... opens the dialog to open a help file in **chm** format. It must have already been created in the project manager under **files/help**. In Runtime, the linked help file is opened if the corresponding alarm is selected and the **Call help** button is clicked. **Note:** The property can only be configured if the In Alarm Message List property is active.

Help chapter	Indication of the help chapter. **Note:** Only available if the `In Alarm Message List` property is activated.
Additional information 1	In the Runtime the additional information entered can be assessed in a VBA macro.
Additional information 2	In the Runtime the additional information entered can be assessed in a VBA macro.
AML/CEL	
In Alarm Message List	`Active`: Will be entered in the AML.
To acknowledge	Only available if `In Alarm Message List` is activated. `Active`: Must be acknowledged.
Comment required	To be able to acknowledge the alarm, a comment must be entered beforehand. The user must be authorized to carry out the necessary function.
To delete	Only available if `In Alarm Message List` is activated. `Active`: Must be deleted.
Print	Only available if `In Alarm Message List` is activated. `Active`: Will be printed via the set standard printer.
In Chronological Event List	`Active`: Will be entered in the CEL. **Hint**: If the initial value (the first value that comes from the controller) or the value when Runtime is started already violates the limit value or the Rema status is active as a result, no entry is created in the CEL. Only once the limit violation has been rectified and then is violated again, or the state becomes inactive and then active again, is a CEL entry generated.
Alarm/event group	Allocation to an alarm/event group.
Alarm/event class	Allocation to an alarm/event class.

VALUE

Parameter	Description
`Value`	Definition of the declaration of value:
	▸ any: any change of value violates the limit
	▸ greater: Enter of a limit
	▸ smaller: Enter of a limit
	▸ equal: Enter of a limit
	▸ Area: Enter of an area (from ... to)
`Case sensitive`	On comparing with the limit string differences in capitalization are regarded.
`Regard wildcards`	The limit string can have wildcards. (Wildcards are only allowed as prefix or suffix; e.g. *xxx or xxx*.)
`Limit text`	Text which is displayed when at a limit violation.
`Treat each change of value as new limit violation`	`Active:` Each change in value is displayed as a limit violation.
`Delay`	Time period which the limit violation must last in order for the limit to be active.
`Status text for counter in the mathematics driver`	Assignment of four possible status numbers for a counter (mathdr32.chm::/23818.htm) in the mathematics driver. For this the variable must have a reaction matrix.

Example

Example 1:
In the sequence of execution Capital after minuscule:
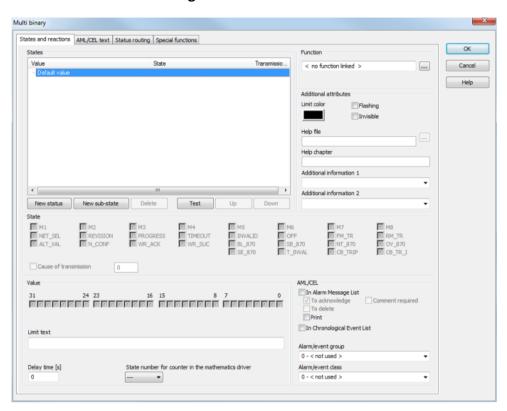Strings: "My", "name", "is", "XY"
Sequence of Execution:
is, Name, My, Henson.

Reason:
is starts with a minusculo, therefore it is higher than the other strings.
Name, N is higher than M and H, My, M is higher than H

Multi-reaction matrices in general

Tabs	Description
States and reactions	Configuration for the states and reactions for the Multi-binary (on page 143) reaction matrix and the Multi-numeric (on page 146) reaction matrix.
AML/CEL text (on page 151)	Configuration for the output of the value next to the limit text in the AML and CEL.
Status routing (on page 151)	Configuration of the status routing.
Special functions (on page 152)	Configuration of special functions.

Note: Status bits NORM and N_NORM are available for both types of multi reaction matrices but not for the standard matrices.

INFORMATION PROCESSING

The upper 16 bits (bit 16 ... 31) are used in the protective data configuring for the information processing; the information number is stored in the upper 16 bits.

The function of the multi-binary reaction matrix is identical to the binary response matrix for the main states. If any of the main states are changed, the respective configuring states (alarm, CEL, text, etc.) are executed:

▶ If a main state has substates, the substates are checked first on reaching the main state.

▶ If no substate corresponds to the current state of the variables, the respective state configuring (alarm, CEL, text, etc.) of the main state is executed.

▶ If the substate corresponds with the current state of the variables, the respective state configuring (alarm, CEL, text, etc.) of the substate is carried out.

▶ If the state of the variables changes from the substate to the main state, there is no new output to the chronologic event list and the alarm list (e.g. breaker tripping treatment in the SAT system).

EXAMPLE：

Parameter	Description
Main value	Engineering -> Lower valuexxxxxx00 no alarm, CEL text off -> xxxxx100 alarm, CEL text Breaker tripping off xxxxxx01 no alarm, CEL text On xxxxxx10 alarm, CEL text Difference xxxxxx11 alarm, CEL text Malfunction
Change from "ON" to "OFF"	no alarm, CEL entry "on"
Change from "ON" to "OFF"	Alarm entry Breaker tripping off, CEL text Breaker tripping off -> Acknowledgement of the breaker tripping (change of BREAKER TRIPPING OFF to OFF) cleared alarm breaker tripping off, no renewed CEL text

For the output in the CEL and in the process screens the entire contents of the variable can be separated to two parts. One part is used for the limit definition, the second part (value) is used in the screens. (Field of application: protection telegrams in SAT 1703).

CAUSE OF TRANSMISSION

Multi-binary and multi-analog reaction matrixes allow evaluation of the transfer cause accordingly

The IEC 60870-5-101/104 standard. The transmission cause consists of:

Bit	Description
6 bit	Value of the cause of transmission 0 ... 63
1 bit	N_CONF (P/N-Bit): 0 = positive 1 = negative confirmation
1 bit	Test bit

In the reaction matrix, the identification of the transmission cause + the N_CONF Bit (P/N bit) can be assigned to each status. So the status is only violated, if the received transmission cause is the same as the defined transmission cause.

Parameter	Description
Transfer cause checkbox	Activates the evaluation of the transfer cause input field.
Transfer cause input field	Expects a value of $0 - 63$ as identification of the transfer cause.

Multi-binary states and reactions

A multi-binary reaction matrix is used for the evaluation of value, state and status bits of digital variables. The lower bits (bit 0.15) are used as value for the display in screens (dynamic elements) and in the chronologic events list (field "value"). In addition to the known configurations of value and status combinations, further configurations are possible.

Parameter	Description
States	List of the engineered states with value, state and cause of transmission.
New status	Creates a new status based on the currently selected state.
Sub-status	Only for multi-binary: Creates a new sub-status for the selected status.
Delete	Deletes all selected statues from the list.
Test	Opens the dialog (on page 154) for testing the status.
Up	Moves selected states up.
Down	Moves selected states down.
Status	Selection of the states.
Cause of Transmission	Additional status for cause of transmission.
Function	Opens the dialog for linking a function. You can deselect an already linked function with the help of button **No selection** in this dialog.
Call via the Alarm Message List	Only available if you activated option In Alarm Message List in **AML/CEL**. Active: Function is called via the AML.
Additional attributes	
Limit color	Color when a limit has been violated.
Flashing	Active: Flashes when a limit has been violated.
Invisible	Active: Will be switch to invisible when a limit has been violated.
Help file	Clicking on ... opens the dialog to open a help file in **chm** format. It must have already been created in the project manager under **files/help**. In Runtime, the linked help file is opened if the corresponding alarm is selected and the **Call help** button is clicked. Note: The property can only be configured if the In Alarm Message List property is active.

Help chapter	Indication of the help chapter. **Note:** Only available if the `In Alarm Message List` property is activated.
Additional information 1	In the Runtime the additional information entered can be assessed in a VBA macro.
Additional information 2	In the Runtime the additional information entered can be assessed in a VBA macro.
AML/CEL	
In Alarm Message List	`Active`: Will be entered in the AML.
To acknowledge	Only available if `In Alarm Message List` is activated. `Active`: Must be acknowledged.
Comment required	To be able to acknowledge the alarm, a comment must be entered beforehand. The user must be authorized to carry out the necessary function.
To delete	Only available if `In Alarm Message List` is activated. `Active`: Must be deleted.
Print	Only available if `In Alarm Message List` is activated. `Active`: Will be printed via the set standard printer.
In Chronological Event List	`Active`: Will be entered in the CEL. **Hint**: If the initial value (the first value that comes from the controller) or the value when Runtime is started already violates the limit value or the Rema status is active as a result, no entry is created in the CEL. Only once the limit violation has been rectified and then is violated again, or the state becomes inactive and then active again, is a CEL entry generated.
Alarm/event group	Allocation to an alarm/event group.
Alarm/event class	Allocation to an alarm/event class.

CHECKBOX STATUS

Each status can have one of 5 possible states. They are defined by clicking in the checkbox before the respective status.

Possible states:

- ▶ `not considered` (period)

- ▶ `on` (1)

- ▶ `off` (0)

- ▶ `coming` (arrow upwards)

- ▶ `going` (arrow downwards)

- ▶ `alternating` (double arrow)

VALUE

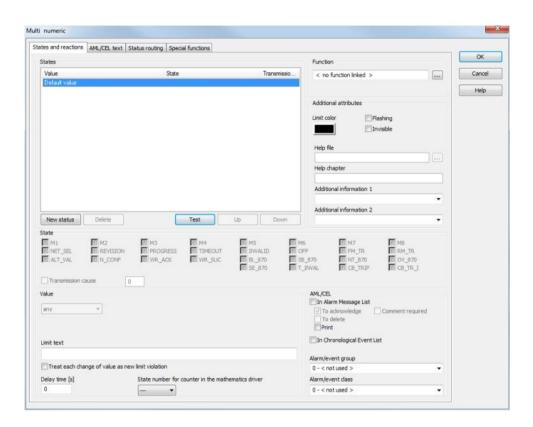| Parameters	Description
`Value`	Current value of variable.
`Limit value text`	Text which is displayed when at a limit violation.
`Delay`	Time period which the limit violation must last in order for the limit to be active.
`Status text for counter in the mathematics driver`	Assignment of four possible status numbers for a counter (mathdr32.chm::/23818.htm) in the mathematics driver. For this the variable must have a reaction matrix.

> ### 💡 Info
>
> *A decimal value can be entered with either a `comma` or a `point` as a decimal separator, it will automatically be changed to a `point`.*

Multi-numeric states and reactions

A `multi-numeric reaction matrix` is used for the evaluation of limit values and status bits of the variables. In addition to the known configurations of limit value and status combinations, further configurations are possible.

Parameter	Description
`States`	List of the engineered states with value, state and cause of transmission.
`New status`	Creates a new status based on the currently selected state.
`Sub-status`	Only for multi-binary: Creates a new sub-status for the selected status.
`Delete`	Deletes all selected statues from the list.
`Test`	Opens the dialog (on page 154) for testing the status.
`Up`	Moves selected states up.
`Down`	Moves selected states down.
`Status`	Selection of the states.
`Cause of Transmission`	Additional status for cause of transmission.
Function	Opens the dialog for linking a function. You can deselect an already linked function with the help of button `No selection` in this dialog.
Call via the Alarm Message List	Only available if you activated option In Alarm Message List in `AML/CEL`. Active: Function is called via the AML.
`Additional attributes`	
Limit color	Color when a limit has been violated.
Flashing	Active: Flashes when a limit has been violated.
Invisible	Active: Will be switch to invisible when a limit has been violated.
Help file	Clicking on ... opens the dialog to open a help file in `chm` format. It must have already been created in the project manager under `files/help`. In Runtime, the linked help file is opened if the corresponding alarm is selected and the `Call help` button is clicked. Note: The property can only be configured if the In Alarm Message List property is active.

Help chapter	Indication of the help chapter. **Note:** Only available if the `In Alarm Message List` property is activated.
Additional information 1	In the Runtime the additional information entered can be assessed in a VBA macro.
Additional information 2	In the Runtime the additional information entered can be assessed in a VBA macro.
AML/CEL	
In Alarm Message List	`Active`: Will be entered in the AML.
To acknowledge	Only available if `In Alarm Message List` is activated. `Active`: Must be acknowledged.
Comment required	To be able to acknowledge the alarm, a comment must be entered beforehand. The user must be authorized to carry out the necessary function.
To delete	Only available if `In Alarm Message List` is activated. `Active`: Must be deleted.
Print	Only available if `In Alarm Message List` is activated. `Active`: Will be printed via the set standard printer.
In Chronological Event List	`Active`: Will be entered in the CEL. **Hint**: If the initial value (the first value that comes from the controller) or the value when Runtime is started already violates the limit value or the Rema status is active as a result, no entry is created in the CEL. Only once the limit violation has been rectified and then is violated again, or the state becomes inactive and then active again, is a CEL entry generated.
Alarm/event group	Allocation to an alarm/event group.
Alarm/event class	Allocation to an alarm/event class.

VALUE

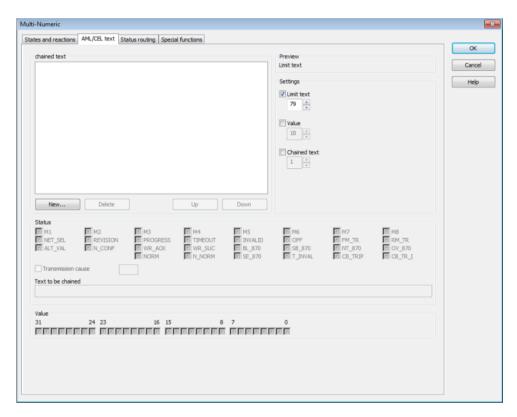| Parameter	Description
Value	Definition of the declaration of value: ▸ any: any change of value violates the limit ▸ greater: Enter of a limit ▸ smaller: Enter of a limit ▸ equal: Enter of a limit ▸ Area: Enter of an area (from ... to)
Limit text	Text which is displayed when at a limit violation.
Treat each change of value as new limit violation	Active: Each change in value is displayed as a limit violation.
Delay	Time period which the limit violation must last in order for the limit to be active.
Status text for counter in the mathematics driver	Assignment of four possible status numbers for a counter (mathdr32.chm::/23818.htm) in the mathematics driver. For this the variable must have a reaction matrix.

Info

A decimal value can be entered with either a `comma` *or a* `point` *as a decimal separator, it will automatically be changed to a* `point`*.*

Configuration AML/CEL text

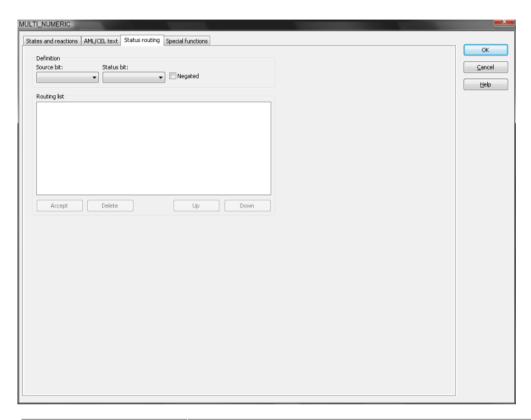In this tab the output for AML and CEL is configured.

Configuration status routing

In this tab the status routing is configured. Routes bits (also negated ones) to other bits.

Possible sources: Value bit 0 ... 31I

Parameters	Description
Source bit	Select source bit from list box
NEG	Negate source bit during the routing
Status bit	Select target bit from list box
Accept	Accept selected routing
Delete	Delete selected routing; Beware: There is no further query
Up	Move selected routing in the sequence upwards
Down	Move selected routing in the sequence downwards

Configuration special functions

In this tab special functions are configured.

With active status bits variables can be switched off automatically or status changes (for reset) can be let pass at switched off variables. For details see chapter Status definition.

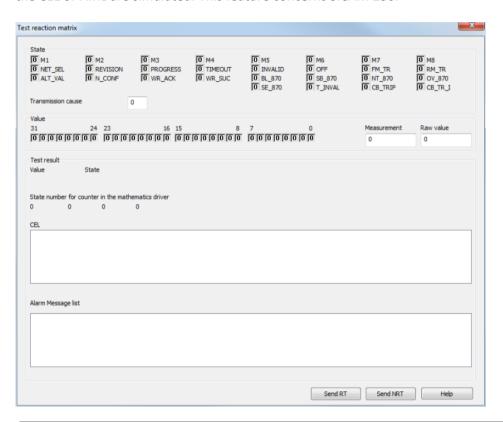| Parameters	Description
switch off for	status bits where the data point is to be switched off Check for "0" or "1" is possible
allow to pass for	status bits where a change in status is to be allowed to pass
No old/new comparison	No monitoring of changes
No old/new comparison for ET-route	No monitoring of changes on the real-time route

Test

Makes it possible for settings for the reaction matrix to be tested. Depending on the configuration of the reaction matrix, a simulation of what happens when defined values are reached is carried out. Entries in the CEL or AML are simulated. This feature concerns SICAM 230.

Parameters	Description
Status	To simulate status changes.
Value	Simulation of value changes.
Test result	Output of changes from status and value in CEL and AML.
SendenEZ	Send changes in real time.
SendenNEZ	Send changes.

2.10.4   Dynamic limit texts in reaction matrices

Dynamic limit texts can also be displayed in reaction matrices:

$ %PV
$ @text

See also Dynamic limit text (on page 117).

Reaction matrices can be used for multiple variables.

2.10.5 Status for value change and delay time

A status can generate alarms depending on values and delay time. For example, an alarm can be created if a variable has not changed for a certain time.

Configuration:

▶ `Value:` `any`

▶ `Treat each change of value as new limit violation:` **acive**

▶ `Delay:` `>0`

In this case:

▶ the alarm is activated if the value is a longer constant than the `delay time`.

▶ If the value changes within the `delay time`, the `delay time` restarts. For example, an alarm can be created if a variable has not changed for a certain time.

2.10.6   List of status bits

Bit number	Short term	Long name	zenon Logic label
0	M1	User status1	_VSB_ST_M1
1	M2	User status2	_VSB_ST_M2
2	M3	User status3	_VSB_ST_M3
3	M4	User status4	_VSB_ST_M4
4	M5	User status5	_VSB_ST_M5
5	M6	User status6	_VSB_ST_M6
6	M7	User status7	_VSB_ST_M7
7	M8	User status8	_VSB_ST_M8
8	NET_SEL	Select in the network	_VSB_SELEC
9	REVISION	Revision	_VSB_REV
10	PROGRESS	In operation	_VSB_DIREC
11	TIMEOUT	Runtime exceedance	_VSB_RTE
12	MAN_VAL	Manual value	_VSB_MVALUE
13	M14	User status14	_VSB_ST_14
14	M15	User status15	_VSB_ST_15
15	M16	User status16	_VSB_ST_16
16	GI	General query	_VSB_GR
17	SPONT	Spontaneous	_VSB_SPONT
18	INVALID	Invalid	_VSB_I_BIT
19	T_CHG_A	Daylight saving time/winter time announcement	_VSB_SUWI
20	OFF	Switched off	_VSB_N_UPD
21	T_EXTERN	Real time external	_VSB_RT_E
22	T_INTERN	Realtime internal	_VSB_RT_I
23	N_SORTAB	Not sortable	_VSB_NSORT

24	FM_TR	Error message transformer value	_VSB_DM_TR
25	RM_TR	Working message transformer value	_VSB_RM_TR
26	INFO	Information for the variable	_VSB_INFO
27	ALT_VAL	Alternate value If no value was transferred, the defined alternate value is used otherwise the last valid value is used.	_VSB_AVALUE
28	RES28	Reserved for internal use (alarm flashing)	_VSB_RES28
29	N_UPDATE	Not updated	_VSB_ACTUAL
30	T_STD	Standard time	_VSB_WINTER
31	RES31	Reserved for internal use (alarm flashing)	_VSB_RES31
32	COT0	Cause of transmission bit 1	_VSB_TCB0
33	COT1	Cause of transmission bit 2	_VSB_TCB1
34	COT2	Cause of transmission bit 3	_VSB_TCB2
35	COT3	Cause of transmission bit 4	_VSB_TCB3
36	COT4	Cause of transmission bit 5	_VSB_TCB4
37	COT5	Cause of transmission bit 6	_VSB_TCB5
38	N_CONF	Negative acceptance of Select by device (IEC 60870)	_VSB_PN_BIT
39	TEST	Test bit (IEC870 [T])	_VSB_T_BIT
40	WR_ACK	Writing acknowledged	_VSB_WR_ACK
41	WR_SUC	Writing successful	_VSB_WR_SUC
42	NORM	Normal status	_VSB_NORM
43	N_NORM	Deviation normal status	_VSB_ABNORM
44	BL_870	IEC 60870 Status: blocked	_VSB_BL_BIT
45	SB_870	IEC 60870 Status: substituted	_VSB_SP_BIT

46	NT_870	IEC 60870 Status: not topical	_VSB_NT_BIT
47	OV_870	IEC 60870 Status: overflow	_VSB_OV_BIT
48	SE_870	IEC 60870 Status: select	_VSB_SE_BIT
49	T_INVAL	Time invalid	not defined
50	CB_TRIP	Breaker tripping detected	not defined
51	CB_TR_I	Breaker tripping detection inactive	not defined
52	RES52	reserved	not defined
53	RES53	reserved	not defined
54	RES54	reserved	not defined
55	RES55	reserved	not defined
56	RES56	reserved	not defined
57	RES57	reserved	not defined
58	RES58	reserved	not defined
59	RES59	reserved	not defined
60	RES60	reserved	not defined
61	RES61	reserved	not defined
62	RES62	reserved	not defined
63	RES63	reserved	not defined

Info

In formulas all status bits are available. For other use the availability can be reduced.

You can read details on status processing in the Status processing chapter.

2.11   Functions for variables

Functions make it possible to start commands in Runtime using a button or script.

To create a function:

1. select **New function**... in the context menu or in the tool bar

2. Navigate to the **Variable**

3. select the desired function

2.11.1   Export data

This function makes it possible to export variable data in Runtime.

The export format and the file name are defined in the Editor. The storage location when executed in Runtime is the `Export` folder in the Runtime folder.
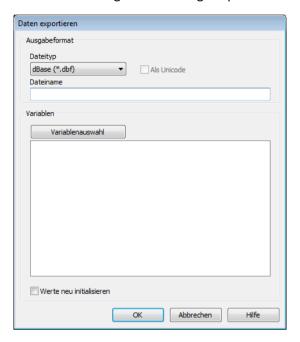
Tip: You can get to the project folder by highlighting the project in the Editor and pressing the key combination `Ctrl+Alt+R`.

To create the function:

1. Create a new function (on page 158)

2. Select **Export data**

3. The configuration dialog is opened

Parameter	Description
Output format	
Format	Select one of the possible file types: ▸ dBase IV (*.dbf) ▸ XML (*.xml) ▸ CSV (*.txt)
As Unicode	This checkbox is only active if you have selected CSV (*.txt) as the `file type`. `Active`: File is saved in Unicode (UTF-16).
File names	Create the filenames. A maximum of 8 characters are permitted for DBF files. Note: The name can no longer be changed in Runtime. Runtime storage location: `Export` folder in Runtime folder.
Variables	
Variable selection	Selection of variables to be exported. A click on the button opens the dialog to select variables. This selection can no longer be changed in Runtime. Note: For a variable to be selectable, you must activate property `Harddisk data storage active` in group `Harddisk data storage` for this variable.
Initialize values again	For the HD values, a ring buffer is filled and stored. `Active`: The ring buffer is emptied and reinitialized for the export. `Inactive`: The ring buffer is kept for the export:

2.11.2   Read a dBase-file

This function is used to read in a DBase file again and issue it as a recipe during online operation.

Give the dBase file as the transfer parameter. The file and directory name have to be DOS compatible.

This file contains the process variable names and the according set values. If the defined variable exists in the project, the value defined in the dBase file is written to the PLC as a new set value.

The dBaseIV file must have the following structure and contents:

Description	Type	Value	Comment
NAME	Char	32	process variable name
WERT	Num	20	Technical value
TYP	Char	8	Data type (C=String, N=Numeric)
DIRECTION	Char	12	

2.11.3   Print current values

This function is used to create a simple log of current variable values (process and derived variables) during online operation.

Give a formatting file as the transfer parameter. This function is configured via an input dialog.

Configurable options are:

Parameters	Description
System buttons	OK, Cancel, Help
With dialog cancel	displays abort dialog during generation in online operation
Select format file	Opens a dialog box for the selection of the *.FRM file. This file is created with the button below. The format FRM is described further below in this chapter.
Start format editor	starts text editor for creation of file
Font	sets font which will be used

Select the file (*.FRM file type by clicking on the "Select format file" button.

Defined keywords are used in the format file. The corresponding variable parameters are entered during online operation instead of the keywords.

Keywords:

Keyword	Description
@TTA.name	outputs variable names
@TTA.wert	outputs current technical value; outputs binary status as value or string variable
@TTA.einheit	outputs unit defined for process variable
@TTA.text	outputs text defined for process variable
@TTA.status	outputs current status bit of process variables in succession with short names (from ZENON6.INI)

The following keywords are available for formatting the output page or as general information:

Keyword	Description
%DATE	outputs current date
%TIME	outputs current time
%PAGE	outputs current page number

Texts for the header and footer can be defined. The sections (header, main section, footer) have to be marked in front and behind the section with "%%". Three empty lines have to be entered behind the last "%%".

Using tabs will result in problems when outputting texts of different lengths. Only empty spaces can be used for positioning The "@" key character is not counted.

FRM FILE "PROJEKT.FRM"

%%
Page header
%%
%%
%date Measuring value log of equipment XYZ %time o' clock

Identification techn. value unit of measurement
Value 1 @Value1.wert bar
Value 2 @VALUE2.wert kV
 Value 3 @VALUE3.wert ºF

Page %page
%%
%%
Page footer
%%

ONLINE PRINTING WITH THE "PROJEKT.FRM" FILE

Header text

21.02.1995 Measuring value log of equipment XYZ 12:00 o' clock

Identification techn. value unit of measurement
Value 1 0.63 bar
Value 2 9.98 kV
> Value 3 17.3 ºF

> Page 5

Footer text

If the desired format of the current values is narrower as the space between the TTA key words, a QRF file with the same name can be used. In the QRF file the process variables are assigned to defined markers. Each marker (xxx) begins with a "$" and is defined with a certain syntax.

`define $xxx as` `Variable`	

– xxx	freely chosen identifier or number
Variable	Variable name

These markers can be accessed in the FRM file The variable parameters are then addressed via other parameters.

$xxx.1	process variable name
$xxx.2	Technical value
$xxx.3	Unit
$xxx.4	Status text

Note: Variable names with a period must be set in apostrophes, for example: "Variable.Test".

Example of a QRF file's structure:

QRF FILE "MOMENT.QRF"

define $001 as TTA1
define $002 as TTA2

FRM FILE "MOMENT.FRM"

%%
Header text
%%
%%
Main section
Value of TTA1=   @$001.2
Value of TTA2= @$002.2
%%
%%
Footer text
%%

ONLINE PRINTOUT WITH THE "MOMENT.FRM" FILE

Header text
Value of TTA1=   100.25
Value of TTA2= 25.745
Footer text

2.11.4   HD administration active

This function is used to activate HD administration (filing of the online values with configured HD feature; online trend) during online operation.

No transfer parameters are needed.

2.11.5   HD administration inactive

This function is used to deactivate HD administration (filing of the online values with configured HD feature ; online trend) during online operation.

> ⚠ **Attention**
>
> *No more values are saved! Online values are updated. No transfer parameters are needed.*

2.11.6   Trend-values on/off

This function is used to switch HD administration (filing of the online values with configured HD feature; online trend) between the two states during online operation. The states stored in the file `project.ini` will be read in when online operation is started.

[DEFAULT]	
...	
HDDATEN=	0 - inactive
	1 - active

2.11.7   Setting values

This function is used to set a set value for a variable in online operation.

The variable, the set value and the way how to set the value have to be given as transfer parameters. This function is configured via an input dialog. The function is configured with its own dialogs for:

▶   numerical variables (on page 168)

▶   binary variables (on page 170)

▶   string variables (on page 173)

The set value dialog box is used for input in Runtime. If the keyboard screen SETVALUEKBD is available in the project, it is used automatically.

🔅 Info

In case of reload or Server-Standby Switch, the present responses or writing affirmations are distorted.

for numeric variables

Parameters	Description
Set value settings	Configuration of the target value.
Set value	Input of the set value to be set for the variable.
Minimum	Input of the minimum value.
Maximum	Input of the maximum value.
Use set value limits of the variable	Active: The set value limits are taken from the variable definition (**Description/Set value**). Inactive: The set value limits can directly be defined in the function (**Minimum** and **Max**).
Options	Configuration of the options.
Propose current value	Active: The value is read from the hardware and then is proposed as a default or directly sent to the hardware. Additionally the caption of the field **Set value** changes to **Change by** and the value entered there is added to the default (subtracted if negative value). Inactive: The value defined under **Set value** is proposed as a default or directly sent to the hardware.
Direct to the hardware	Active: After activating the function the value is directly sent to the hardware not asking for a further confirmation. Inactive: Before sending the value to the hardware the set value dialog is opened and the user can change the settings in the Runtime.
Wait for writing confirmation	Active: The function only finshes execution once a positive confirmation of the write action is received or the fixed defined timeout of 30 seconds is over. In scripts this can make sure, that the following function is not started, before this one is finished. The confirmation can also be evaluated with the status bit **WR-SUC**. Inactive: Do not wait for the writing confirmation.
Write set value via	Configuration of how the target value is to be set: ▸ Standard dialog box: Set value in Runtime set using a dialog box. ▸ Screen: Set value in Runtime set using a screen. ▸ Program: Set value in Runtime set using a program. Selection

	by clicking on the `...` button.
OK	Configuration is accepted, function created and the dialog is closed.
Cancel	Configuration is rejected, function created without parameters and the dialog is closed.
Help	Online help is opened.

💡 **Info**

A decimal value can be entered with a colon as well as with a point, the decimal point will automatically be changed to a point.

for binary variables

Parameters	Description
LOW (0)	Value of variables will be set to LOW (0)
HIGH (1)	Value of variables will be set to HIGH (1)
TOGGLE	Value of variable is switched to complementary status. Hint: If the variable value is to be toggled, we recommend activating the DDE active property. Otherwise Runtime waits until the driver knows the value of the variables. If the value is not available due to a breakdown in communication with the control unit, the waiting time corresponds to the timeout time of the driver.
Parameters	Description

Set value settings	Configuration of the target value.
`Set value`	Input of the set value to be set for the variable.
`Minimum`	Input of the minimum value.
`Maximum`	Input of the maximum value.
`Use set value limits of the variable`	`Active:` The set value limits are taken from the variable definition (**Description/Set value**). `Inactive:` The set value limits can directly be defined in the function (**Minimum** and **Max**).
Options	Configuration of the options.
`Propose current value`	`Active:` The value is read from the hardware and then is proposed as a default or directly sent to the hardware. Additionally the caption of the field **Set value** changes to **Change by** and the value entered there is added to the default (subtracted if negative value). `Inactive:` The value defined under **Set value** is proposed as a default or directly sent to the hardware.
`Direct to the hardware`	`Active:` After activating the function the value is directly sent to the hardware not asking for a further confirmation. `Inactive:` Before sending the value to the hardware the set value dialog is opened and the user can change the settings in the Runtime.
`Wait for writing confirmation`	`Active:` The function only finshes execution once a positive confirmation of the write action is received or the fixed defined timeout of 30 seconds is over. In scripts this can make sure, that the following function is not started, before this one is finished. The confirmation can also be evaluated with the status bit **WR-SUC**. `Inactive:` Do not wait for the writing confirmation.
`Write set value via`	Configuration of how the target value is to be set: ▸ `Standard dialog box:` Set value in Runtime set using a dialog box. ▸ `Screen:` Set value in Runtime set using a screen. ▸ `Program:` Set value in Runtime set using a program. Selection by clicking on the `...` button.
OK	Configuration is accepted, function created and the dialog is closed.
Cancel	Configuration is rejected, function created without parameters and the dialog is closed.

Help	Online help is opened.

for string variables

Parameters	Description
Set value settings	Configuration of the target value.
Set value	Input of the set value to be set for the variable.
Options	Configuration of the options.
Propose current value	Active: The value is read from the hardware and then is proposed as a default or directly sent to the hardware. Additionally the caption of the field **Set value** changes to **Change by** and the value entered there is added to the default (subtracted if negative value). Inactive: The value defined under **Set value** is proposed as a default or directly sent to the hardware.
Direct to the hardware	Active: After activating the function the value is directly sent to the hardware not asking for a further confirmation. Inactive: Before sending the value to the hardware the set value dialog is opened and the user can change the settings in the Runtime.
Wait for writing confirmation	Active: The function only finshes execution once a positive confirmation of the write action is received or the fixed defined timeout of 30 seconds is over. In scripts this can make sure, that the following function is not started, before this one is finished. The confirmation can also be evaluated with the status bit **WR-SUC**. Inactive: Do not wait for the writing confirmation.
Write set value via	Configuration of how the target value is to be set: ▸ Standard dialog box: Set value in Runtime set using a dialog box. ▸ Screen: Set value in Runtime set using a screen. ▸ Program: Set value in Runtime set using a program. Selection by clicking on the ... button.
OK	Configuration is accepted, function created and the dialog is closed.
Cancel	Configuration is rejected, function created without parameters and the dialog is closed.
Help	Online help is opened.

Check write set value

When writing values, the value receives a status bit that is has been written. If the writting process is successful, the corresponding status bit is set:

▶   WR-ACK

The driver received a value for writing.

▶   WR-SUC

Value `1`: Writing successful.

Value `0`: Writing not successful. The value could not be written.

> 💡 **Info**
>
> *In case of reload or Server-Standby switch, the currently active responses or writing affirmations are discarded.*

This status combination are active until the next value change is triggered. Then both states are set to `0` until the writing action is finished. For evaluation the following bit combination must be requested in the reaction matrix:

`WR-ACK, WR-SUC`

Result:

▶   WR-ACK `1`, WR-SUC `1`: Writing action successful.

▶   WR-ACK `1`, WR-SUC `0`: Writing action not successful.

> ⚠ **Attention**
>
> *The mechanism only shows, that the writing action was successful (or not successful) to the PLC. This does not mean, that the value has indeed been changed in the PLC, since the PLC can reset/overwrite the value immediately. (For example for writing the outputs or the transient bits which are only set for a short time.)*

MODULES

This mechanism can be used in the following modules:

- ► function **Write set value** (on page 167): Activate option `Wait for writing confirmation` in the configuration dialog of the function.

- ► **Standard recipes**: Activate property `Write synchronously` .

- ► **Recipegroup Manager**: Activate property `Write synchronously`.

ENTRY IN CEL

- ► Function Write set value

For the entry in the CEL you must activate property `Function Set SV` in node `Chronologic event list` in the project settings. After this the positive or negative response the execution of the function is written to the CEL.

- ► Standard recipes and Recipegroup Manager

For the entry in the CEL a system driver variable is used which is set to `1` when a recipe is written successfully. A global variable is evaluated on the Server, a local variable on every Client in order to determine when the recipe executed last was written completely.
With this variables a CEL entry can be created via limit or reaction matrix (on page 122). The query is carried out via a multi analog (on page 146) or a multi binary (on page 143) reaction matrix.
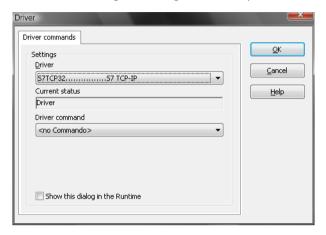
2.11.8 Driver commands

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.

2.11.9   Write time to variable

With this function a connected PLC can be synchronized with the PC. When executing this function, the System time is taken and saved in a String varialbe of the format: DD.MM.YYYY HH:MM:SS

The name of the variable has to be given as a parameter.

2.11.10 Read time from variable

With this function the PC can be synchronized with a connected PLC. On executing the function a string variable is read from the PLC. If the difference between PC and PLC time is larger than one second, the PC system time is set and an entry in the CEL is generated. The string variable must have the following format: `DD.MM.YYYY HH:MM:SS`.

Note: Time is saved as local time. For details see chapter Handling of date and time in chapter Runtime.

The name of the variable has to be given as a parameter.

2.12 Screen Variable diagnosis

By using screen `Variable diagnosis`, variables can be displayed and set values can be written in the Runtime. You will find more information on the pre-defined screen types in the chapter 'Screens / Screen types'.)
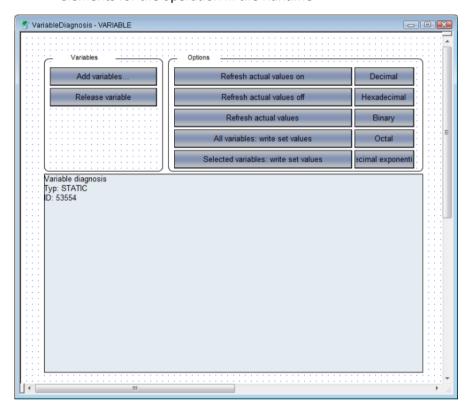
2.12.1 Create screen Variable diagnosis

In order to create screen `Variable diagnosis`:

▶ create a new screen

▶ select `Variable diagnosis` as screen type

▶ an empty screen is created

▶   select the menu item **Add template** from menu **Control elements** or select the desired control elements for the operation in the Runtime

Control element	Description
`Add template`	Opens the dialog for selecting a template for the screen type. Templates are shipped together with zenon and can also be created by the user. Templates add pre-defined control elements to pre-defined locations in the screen. Elements that are not necessary can also be removed individually once they have been created. Additional elements are selected from the drop-down list and dragged onto the screen. Elements can be moved on the screen and arranged individually.
List view	
`Variable diagnosis`	List field in the Runtime.
`Values decimal`	The variable values in the list (actual value, set value) are displayed decimal.
`Values hexadecimal`	The variable values in the list (actual value, set value) are displayed hexadecimal.
`Values binary`	The variable values in the list (actual value, set value) are displayed binary.
`Values octal`	The variable values in the list (actual value, set value) are displayed octal.
`Values decimal exponential`	The variable values in the list (actual value, set value) are displayed decimal exponential.
Variables	
`Add variable`	Add a variable to the list in the Runtime
`Release variable`	Remove variable from the list in the Runtime
Update value	
`Updating actual value ON`	Switch on automatic updating of current variable values.
`Updating actual value OFF`	Switch off automatic updating of current variable values.
`Refresh actual values`	Update the current values of the variables.

`All variables: Write set value`	Write the set values of all variables to the hardware.
`Selected variables: Write set value`	Write the set values of the selected variables to the hardware.
`Change set value`	Edit the set value of the selected variable.
Special operations	
`Selected variables: Switch to spontaneous value`	Switch selected variable to spontaneous value.
`Selected variables: Switch to alternate value`	Switch selected variable to alternate value.
`Selected variables: Modify alternate value`	Modify the alternate value of the selected variable.
`Selected variables:  Switch on spontaneous value`	Switch on the spontaneous value for the selected variable. To do this, the OFF status bit (bit 20) is reset.
`Selected variables:  Switch off spontaneous value`	Switch off the spontaneous value for the selected variable. To do this, the OFF status bit (bit 20) is set.
`Selected variables: Switch on revision`	Switch on the revision for selected variables. To do this, the REVISION status bit (bit 09) is set.
`Selected variables: Switch off revision`	Switch off the revision for selected variables. To do this, the REVISION status bit (bit 09) is reset.
Filter profiles...	
`Profile selection`	Select saved profile
`Save`	Save settings as profile
`Delete`	Delete profile

💡 Info

A decimal value can be entered with either a `comma` or a `point` as a decimal separator, it will automatically be changed to a `point`.

In order to use the variable diagnosis in the Runtime, create a function screen switch (on page 183) to the screen.

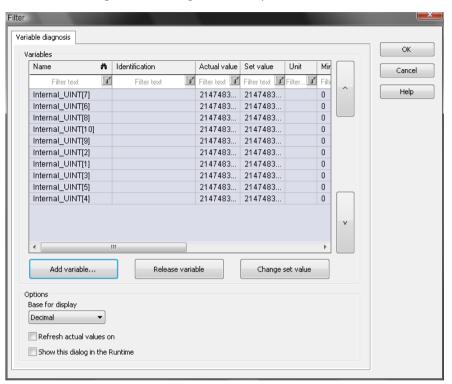## 2.12.2   Screen switch - Variable diagnosis

The parameters needed for opening a screen of the type `Variable diagnosis` are the variables to be displayed in the screen and the default set values for the selected variables.

In order to create a function to switch to the screen `Variable diagnosis`:

▶   select New Function

▶   select screen switch

▶   select the screen Variable diagnosis (on page 179)

▶ the dialog for the configuration is opened

Parameter	Description
`Variables`	List of the variables and their properties.
`Add variable`	Opens the dialog for the variable selection in order to add one or more variables.
`Release variable`	Selected variables are released from the screen.
`Change set value`	The default set values for the variables of the screen can be changed. Set values can also be changed by clicking the cell with the set value.
`Options`	
`Base for display`	Defines the display of the values in the Runtime. Possible values: ▸ Decimal ▸ Hexadecimal ▸ Binary ▸ Octal ▸ Decimal exponential
`Refresh actual values on`	`Active:` The actual values are updated when the window is called up in the Runtime.
`Show this dialog in the Runtime`	`Active:` Opens the dialog when the screen is opened in the Runtime.

💡 Info

You can change the columns (displayed text, column type and column width) by right-clicking the column header. You can find the setting for this in the context menu. They are transferred from the Editor to the Runtime and can be changed their in the same name if necessary.

2.13   Unit conversion

Unit conversion enables conversion and switching of base units into conversion units, for example meters into yards or meters into decimeters, centimeters and millimeters. A base unit contains the initial value for a conversion. The conversion unit contains the converted value in relation to the base value. Both have a unit name. A `Factor`, a `Offset` and a `Shift of the decimal point` can be defined for conversion units, based on the relevant base unit.

A base unit can be selected when setting parameters for a variable. It is possible to switch between the different units during runtime using the Unit switching (on page 191) function.

> **License information**
>
> *Part of the standard license of the Editor and Runtime.*

PROJECT MANAGER CONTEXT MENU

Menu item	Action
New base unit	Creates a new base unit.
Export XML all	Exports all entries as an XML file.
Import XML	Imports units from an XML file.
Help	Opens online help.

> **Info**
>
> You must not use the unit conversion together with the variablerw report command. This report command provides a unit conversion for older projects. If the unit defined in the unit conversion of a variable is changed by the report command in Runtime, you must carry out the configuration in the unit conversion again.

2.13.1 Units detail view of toolbar and context menu

CONTEXT MENU UNITS DETAIL VIEW

Menu item	Action
New base unit	Creates a new base unit.
Export XML all	Exports all entries as an XML file.
Import XML	Imports units from an XML file.
Help	Opens online help.

CONTEXT MENU AND TOOL BAR BASE UNIT/CONVERSION UNIT

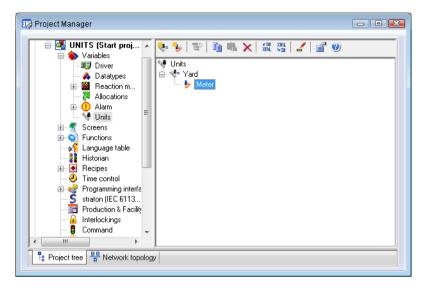| Menu item	Action
New base unit	Creates a new base unit.
New conversion unit	Creates a new conversion unit for the superordinate base unit.
Linked elements: Jump back to starting element	Combobox with link back to the element from which you can reach the unit. Only available if the unit is linked to another element.
Rename	Makes it possible to rename the unit. Attention: If units are renamed, all units that are already linked to variables or used in functions must then have the respective variable or command amended manually. See also: Allocate a base unit to a variable (on page 190) and Unit conversion command (on page 191).
Export XML all	Exports all entries as an XML file.
Import XML	Imports units from an XML file.
Copy	Copies the selected entries to the clipboard.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of".
Delete	Deletes selected entries.
Help	Opens online help.

2.13.2 Engineer units

You must create a base unit and a conversion unit for the conversion, as well as defining the parameters for conversion. The conversion is carried out using the formula $y=kx+d$ (Austria) or $y=mx+b$ (Germany). Each base unit can be allocated to a variable (on page 190) as a unit.

TO CREATE A NEW BASE UNIT:

▶ select the Units node in Project Manager

▶ right-click on Units
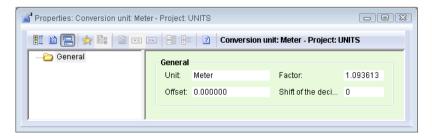
▶ select the New base unit command from the context menu

▶ a new entry is created in the list

▶ give the unit a name

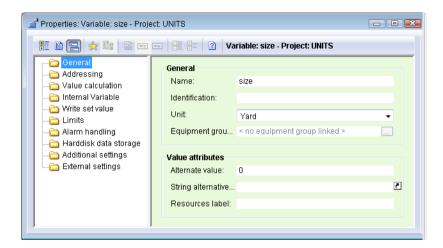▶ create the conversion unit

TO CREATE A NEW CONVERSION UNIT：

▶ right-click on a base unit

▶ select the Conversion unit command from the context menu

▶ a new entry is created in the list

▶ give the unit a name

▶ define a `Factor` for the conversion

▶ define a value for the `Shift of the decimal point`

▶ define a value for the `Offset`
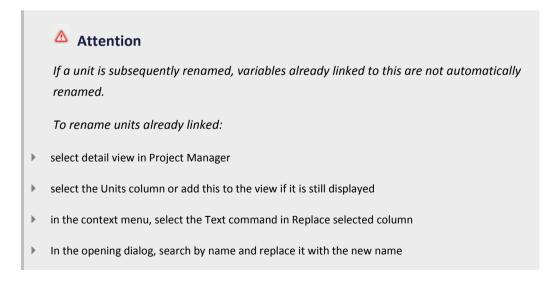
2.13.3 Allocate a base unit to a variable

You allocate a variable to base units in the `Measuring unit` (nodes `General`) property.

You are free to name units as you wish here. If the unit conversion is used during runtime, select a pre-defined basis unit from the drop-down list.

Hint: If you give it a name of your choice, it is best to create a link with the same name straight away in Node units - the basic unit (on page 188).

You must create a Unit conversion function (on page 191) in order to be able to convert during runtime.
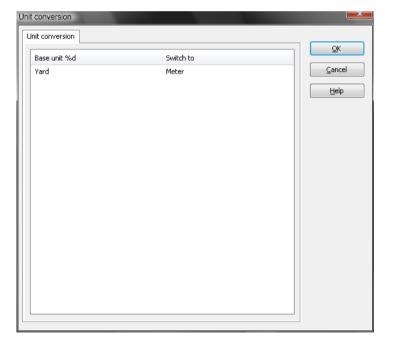
> ⚠ **Attention**
>
> *If a unit is subsequently renamed, variables already linked to this are not automatically renamed.*
>
> *To rename units already linked:*

▸ select detail view in Project Manager

▸ select the Units column or add this to the view if it is still displayed

▸ in the context menu, select the Text command in Replace selected column

▸ In the opening dialog, search by name and replace it with the new name

2.13.4   Function unit conversion

In order to carry out unit conversion in the Runtime, create function unit conversion:

▶   select the Functions node in Project Manager

▶   in the context menu, select the command New function...

▶   navigate to the variable

▶   Select the unit conversion function

Note: The execution of function Unit conversion triggers a refresh of the report if it is displayed at the moment.

▶   The dialog for the definition of the Unit conversion opens.

Property	Description
Unit conversion	Dialog for the allocation of conversion units to basic units.
Base unit	List of the created basic units.
Switch to	Drop-down list for the selection of the conversion unit. You can either select a conversion unit or the basic unit. Engineered conversion unit: In the Runtime the basic unit is converted to the conversion unit. <Base unit>: The basic unit is still active in the Runtime.

⭐ Info

The units are not exported with the XML export of this function. You must export the units separately.

⚠ Attention

If a unit is renamed afterwards, the renamed basic unit is automatically taken into consideration in the function. However you must change the conversion units manually.

2.13.5   Runtime

Each variable value for each input or output in addition to those used as standard when converting signal units to measuring range units is converted in runtime.

▸ Output: A conversion unit for a variable is activated with the Unit switch function (on page 191). The value in measuring units is subject to the pre-defined offset and factor. In addition, to convert the value into a string, the number of decimals set for a variable is corrected accordingly.

▸ Input The conversion is carried out along the lines of output in the other direction.

LIMITATIONS：

▶ At the export the new units are exported.

▶ Values that are saved as a string are not recalculated for output. These values remain in the unit that was active at the time of creating the string. This particularly affects all values inserted into the text of a CEL entry, such as "Set value changed from OLD to NEW" etc.

▶ Operating hours and operations counters in Industrial Maintenance Manager are always displayed in base units here.

▶ Outputs in the EMS screen are always displayed in the base unit.

VBA

Values above VBA are always accessed in base units. For example, **Variable.Value** does not provide a value with units switched, because it is not a value output. 4 new functions have been incorporated into **Variable.** so that unit switching can also be used above VBA:

Keyword	Description
SecondaryUnitName	gives the name of the conversion unit set
SecondaryUnitDigits	gives the decimals for the conversion unit set
CalcSecondaryUnitValue	converts the value of the `base unit` into the value of the `conversion unit`
CalcPrimaryUnitValue	converts the value of the `conversion unit` into the value of the `base unit`