zenon manual
Variables

v.8.20

COPA-DATA

Contents

1 Welcome to COPA-DATA help

ZENON VIDEO TUTORIALS

You can find practical examples for project configuration with zenon in our YouTube channel (https://www.copadata.com/tutorial_menu). The tutorials are grouped according to topics and give an initial insight into working with different zenon modules. All tutorials are available in English.

GENERAL HELP

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

PROJECT SUPPORT

You can receive support for any real project you may have from our customer service team, which you can contact via email at support@copadata.com.

LICENSES AND MODULES

If you find that you need other modules or licenses, our staff will be happy to help you. Email sales@copadata.com.

2 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.

> ### 💡 Information
>
> Variables have to be licensed. The number of licensed TAGs is displayed in the zenon Editor. The variables that are licensed and still available are displayed in the zenon Runtime. If more TAGs are configured or used than are licensed, an info box indicates the available variables.

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 74) on two components: Data type (on page 15) and Driver objekt type. These two components are independent and thus can be configured independently.
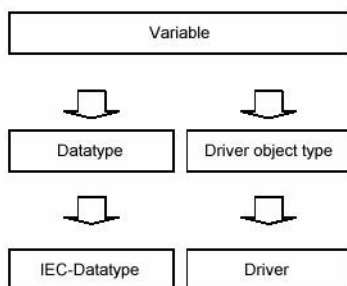
PROJECT MANAGER CONTEXT MENU

Menu item	Action
New variable...	Opens the wizard for creating a new variable.
Export all as XML	Exports all entries as an XML file.
Import XML...	Imports entries from an XML file.
Extended import/export	Opens the menu for the import and export of Step 7 projects. Supported file formats: dBase and CSV. **Note:** For additional information please read the chapter Step 7-project import in the book import - export.
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 with predefined editor profiles.
Help	Opens online help.

STATUS

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

Communication problems are displayed in the Runtime with a status and highlighted in color. Default color coding of screen items for errors:

▶ **Communication failure to PLC**: *red square* (hex value: *#FF0000*)

▶ **Communication failure to server**:*blue square* (hex value: *#0000FF*)

▶ **Unknown value in Process Recorder**: *yellow square* (hex value: *#FFC000*)

These colors can be adjusted in the project properties in the **Graphical design**/**Status of variable** node.

💡 **Information**

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

3 Variables detail view of toolbar and context menu

TOOLBAR

Symbol	Action
New variable	Opens the dialog for creating a new variable.
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.
Variable use	Creates a project analysis for selected variables in the current project and displays it as a result list in its own window.
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.
Jump back to starting element	Jumps back to the initial position in the zenon Editor. **Note:** Only available in the context menu if a jump to the current position has been made from another position with the **Linked elements** context menu entry.

Symbol	Action
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 after a confirmation from list.
Expand/collapse	Drop-down list in order to expand or to collapse all nodes or the selected 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 as XML...	Exports selected elements as an XML file.
Import XML	Imports variables from an XML file.
Import/Export	Opens the menu for the import and export of Step 7 projects. Supported file formats: dBase and CSV. **Note:** For additional information please read the chapter Step 7-project import in the book import - export.
Extended filter	Activate/Deactivate expanded filter Activating opens the dialog to select the filter criteria.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
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 cells that can be edited can be selected.
Replace text in selected column	Opens the dialog to search and replace texts for the selected column.
Replace text in limit values	Opens the dialog for the selection of the **Dynamic properties** Once the property has been selected, the search and replace dialog is opened by clicking on **OK**.
Properties	Opens the **Properties** window.

Symbol	Action
Help	Opens online help.

CONTEXT MENU

Menu item	Action
New variable	Opens the dialog for creating a new variable.
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.
Linked elements	Opens the submenu with linked elements.
Variable use	Creates a project analysis for selected variables in the current project and displays it as a result list in its own window.
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.
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 after a confirmation from list.
Expand/collapse node	Allows all or selected nodes to be expanded or collapsed. Selection: ▶ **Expand all** ▶ **Collapse all** ▶ **Expand selection** ▶ **Collapse selection**
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 as XML...	Exports all selected entries as an XML file.
Import XML	Imports entries from an XML file.
Extended import/export	Opens the menu for the import and export of Step 7 projects. Supported file formats: dBase and CSV.

Menu item	Action
	Note: For additional information please read the chapter Step 7-project import in the book import - export.
Extended filter	Activate/Deactivate expanded filter Activating opens the dialog to select the filter criteria.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
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 cells that can be edited can be selected.
Replace text in selected column	Opens the dialog to search and replace texts for the selected column.
Replace text in limit values	Opens the dialog for the selection of the **Dynamic properties** Once the property has been selected, the search and replace dialog is opened by clicking on **OK**.
Properties	Opens the **Properties** window.
Help	Opens online help.

4 Naming of objects

The name of an object must be unique in zenon.

💡 Information

Note the following when assigning names:

▶   the characters **#** and **@** are not permitted in 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

   ▶ that 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

   > ### ♀ Information
   >
   > Recommendation: Use unique object names which differ from each other not only by case sensitivity.

5 Activating variables in zenon

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

> ### ♀ Information
>
> The switching time of screens in the Runtime depends on the number of linked variables. Only once all variables have been successfully signed in can the screen be operated.
>
> If many variables are operated, this can slow the switching time. In this case, a progress bar is shown, which displays the loading progress.

ACTIVATE (ADVISE):

Advised 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/modify set value** Function (on page 196) 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

▶ Module **Process Recorder**

▶ VBA/VSTA Online Container

▶ Property**Harddisk data storage active** activated

▶ Property**Permanently read variable** activated

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

Variables that are activated via the **Permanently read variable** property are also available to other Windows applications.

Hint: If the variable value is to be toggled or to be saved in the CEL with the new value and the old value, it is recommended that the **Permanently read variable** property is activated.

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.

5.1   Variables in the Runtime

Each variable in zenon contains the following information in the Runtime:

▶ Data type

▶ Value

▶ Raw value (only from PLC)

▶ adjusted value (linear/ non-linear) **Value calculation**

> ▶ normal value (for boolean) **Normal value**

▶ Timestamp

> ▶ internal (64 bit) stamped by the driver always in microseconds

> ▶ external (64 bit) resolution depends on the communication protocol and the PLC

> ▶ Format: UTC - local time

> ▶ Resolution: Microseconds

▶ Status

6 Data Types

Each variable is based on an IEC data type. The data type has the same properties as the variable itself (unit, signal resolution, limit values, 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 34) 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. In doing so, the properties are referenced by the data type on which they are based. That means: If a property of the data type is changed, it is also changed for all variables that have been created with this data type. It is possible to see that a value is referenced from the corresponding symbol (arrow) in the properties window. This reference can also be cut. You can find detailed information on this in the Inheritance concept (on page 110) chapter.

Pre-defined styles or a pre-defined style group can be assigned to a data type, which is then applied in the Runtime in Extended Trend. For detailed information refer to chapter Linking variabels/data types with style group/scale style/curve style (on page 109).

PROJECT MANAGER CONTEXT MENU

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 all as XML	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 moment zenon supports the following IEC data types:

Short name	Long name	Comment / value range	Number of bits
BOOL	Boolean	Bit. Range of values: *0/1 - FALSE/TRUE* **Note:** Limits are not deleted automatically as they may be lost for a wrong variable type selection. For exampel if you change a numeric variable with a defined limit to type *BOOL*, this variable will have three limits.	1
BYTE	Oktett	Arranged compilation of 8 bit. Value range: *0* to *+256*	8
DATE	Date	IEC date: in steps of *1* day. Value range: *1990-1-1* to *2168-12-31*	16
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. Value range: *1990-1-1-0:0:0.0* to *2089-12-31-23:59:59.999*	64
DINT	Double integer	Signed double: Value range: *-2147483648* to *+2147483647*	32
DWORD	Double Word	Dual and hexadecimal figures with *32* bits. Value range: *(0,0,0,0)* to *(255,255,255,255)*	32
INT	Integer	Signed word. Value range: *-32768* to *+32767*	16
LINT	Long integer	Because zenon cannot hold the value	64 or 52

Short name	Long name	Comment / value range	Number of bits
		range of a *64* bit number, the actual value range is restricted to *52* bits. Numbers with signs are possible from *–2251799813685248-2251799813685247*. Numbers outside of this range cause a overflow or underflow, according to a *52*-bit *Integer*. The following areas work with full 64 bit resolution in zenon Runtime: 1. The Shared Memory VBA Interface if the value is given 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. **Note:** RGM and recipes cannot save even a *52* bit value with full accuracy. Value range: *-9223372036854775808* to *+9223372036854775807*	
LREAL	Real numbers	Real numbers Range of values: *-1,7E+308* to *+1,7E+308* Accuracy: 15 digits.	64
REAL	Real numbers	Real numbers The *REAL* data type with single point precision (32-bit) covers a large range of figures, but has limited precision. When converting decimal values to values of the *REAL* data type, it is	32

Short name	Long name	Comment / value range	Number of bits
		possible, depending on the size, that rounding errors occur with the seventh decimal place. The administration of decimal values with more than *6* significant positions is therefore not recommended from a technical point of view. **Recommendation:** Use the *LREAL* data type instead of *REAL* if the PLC supports it. The possible higher precision that would result in comparison to the *REAL* data type is, for example, necessary for counter statuses. If they are values that are counted in whole-number increments, an integer data type is recommended. For example, a *UDINT* variable can save any whole-number value between *0* and *+4.294.967.295* without errors. If counter states go above this limit, usually two *DINT* variables are cascaded or the counter is reset. **Tip/workaround:** If you nevertheless want to continue working with the Data type *REAL* , (such as when there is a limitation of the data types due to the PLC), the bit string of the PLC can be read as *DINT* instead of *REAL*. This bit string can be stripped down in zenon Logic with the *AND_MASK* and *OR_MASK* function blocks and the components of the *REAL* value (sign, exponent, fixed-point part) can be extracted. Exact counter values can be extracted from this. Value range: *-3,4E+38* to *+3,4E+38* Accuracy: 6 digits	

Short name	Long name	Comment / value range	Number of bits
SINT	Short integer	Prefixed *Byte*. Value range: *-128* to *+127*	8
STRING	Variable-length single byte character string	ASCII String. The maximum 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. Range of values: *0* to *+254* characters.	>= 8 x string length
TIME	Duration	Duration in the IEC format. Signed integer IEC time in steps of *1* ms. Range of values: *-24D_20H_31M_23S_648MS* to *24D_20H_31M_23S_647MS*	32
TOD	Time of day	Time of day in steps of *1 ms*. Value range: *0:0:0.0* to *23:59:59.999*	32
UDINT	Unsigned double integer	*Double*: Value range: *0* to *+4294967295*	32
UINT	Unsigned integer	*Word*. Value range: *0* to *+65535*	16
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 *+4503599627370495*. Numbers outside of this range cause a overflow or underflow, according to a 52-bit integer. The following components work with a	64 or 52

Short name	Long name	Comment / value range	Number of bits
		full 64 bit resolution in the zenon Runtime: ‣ The shared memory VBA Interface, when the value was entered in signal resolution. ‣ The shared memory zenon Logic interface. ‣ 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. Value range: *0* to *+18446744073709551615*	
USINT	Unsigned short integer	*Byte*: Value range: *0* to *+255*	8
WORD	Word	Dual and hexadecimal figures with *16* bits. Value range: *(0,0)* to *(255,255)*	16
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. Range of values: *0* to *+16382* characters.	>= N x string length

Note: You can find information on how the inheritance of data type properties works in the Inheritance concept (on page 110) section.

> **⚠Attention**
>
> In the zenon Runtime, the data types *DATE*, *TIME*, *TOD* are treated as a data type *UDINT* in the Unix time format (seconds passed since 01. 01. 1970). The data type *DATE_AND_TIME* equals data type *FLOAT* 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.

6.1 Data types detail view toolbar and context menu

CONTEXT MENU

Menu item	Action
New simple datatype...	Opens the dialog for creating a new simple data type.
New structure datatype...	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.
Activate all variables based on it	Activates all variables that are based on the selected data type.
Move up	Moves the entry up one place in the recipe list. **Note:** Only available for structure elements.
Move down	Moves the entry down one place in the recipe list. **Note:** Only available for structure elements.
Jump back to starting element	Jumps back to the initial position in the zenon Editor. **Note:** Only available in the context menu if a jump to the current position has been made from another position with the **Linked elements** context menu entry.
Copy	Copies the selected entries to the clipboard.

Menu item	Action
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 after a confirmation from list.
Expand/collapse node	Allows all or selected nodes to be expanded or collapsed. Selection: ▶ **Expand all** ▶ **Collapse all** ▶ **Expand selection** ▶ **Collapse selection**
Export selected as XML...	Exports all selected entries as an XML file.
Import XML	Imports entries from an XML file.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
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 cells that can be edited can be selected.
Replace text in selected column...	Opens the dialog to search and replace texts for the selected column.
Replace text in limit values...	Opens the dialog for the selection of the **Dynamic properties** Once the property has been selected, the search and replace dialog is opened by clicking on **OK**.
Properties	Opens the **Properties** window.
Help	Opens online help.

6.2 Types of data types

Basically there a two different kinds of data types:

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

6.2.1 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.

6.2.2 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.

6.2.2.1 Creating a new user-defined datatype

To create a new user-defined data type:

1. Select the **Variables** node.

2. Click on **Data type**.

   All existing data types are listed in the list of 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**.

3. Click on the list with the right mouse button.

4. Select **New data type ...** in the context menu.

   The **Create simple data type** dialog is opened.
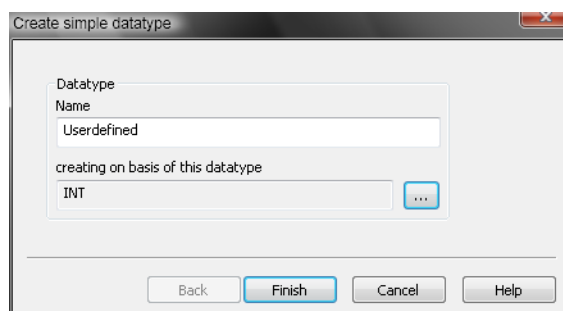
5. Enter the name of the user-defined data type.

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

6. Select a pre-existing data type. This datatype is a template for the new datatype to be created.

7. Click on the **Finish** button.

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

Note:

▶ All properties are copied to the new data type. They can subsequently be changed in the properties window.

▶ No reference to the original data type, from which the copy was made, remains.

▶ You can also use the newly-created file type as a template for further data types.

6.2.2.2 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.

The IEC data type should not be changed as the signal resolution would change.
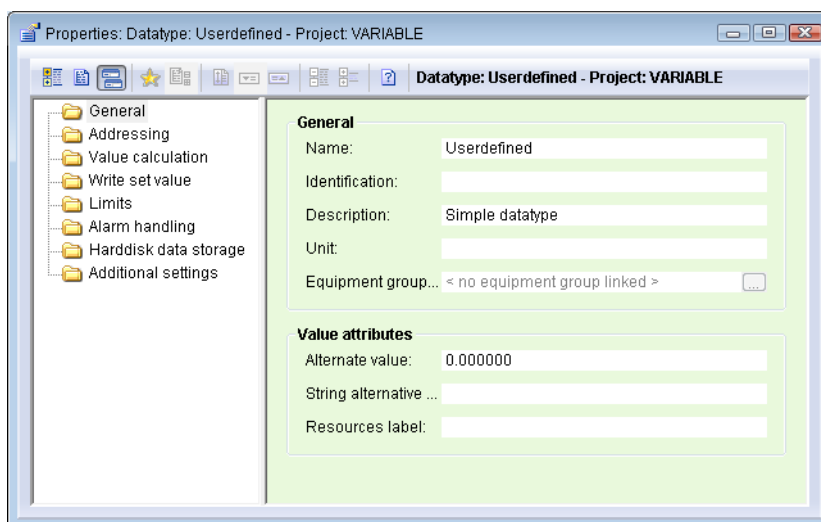
6.2.2.3 Deleting an user-defined datatype

Select the data type to be deleted in the list in the detail view and confirm the delete action in the following dialog box.
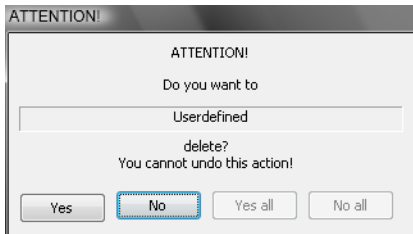
⚠**Attention**

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

6.2.3 Structure data types

Structure data types are always user-defined data types (on page 15). 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**

When creating, it is established whether the structure element should use a *linked* or *embedded* type. Changes afterwards are not possible.

EXAMPLE FOR A STRUCTURE DATATYPE:

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.

6.2.3.1 Creating a structure datatype

This is how you create a structure datatype:
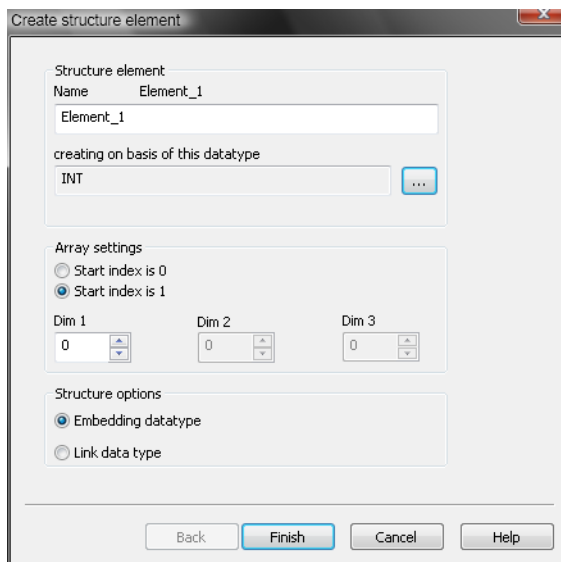
1. Right click on a user-defined datatype in the list and select **New structure datatype** in the context menu.

2. You define the names in the dialog box that now appears.
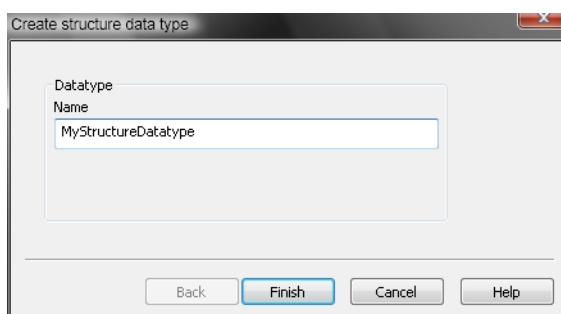
Note: The name of the data type is limited to 128 characters.

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. Please enter a unique 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**.
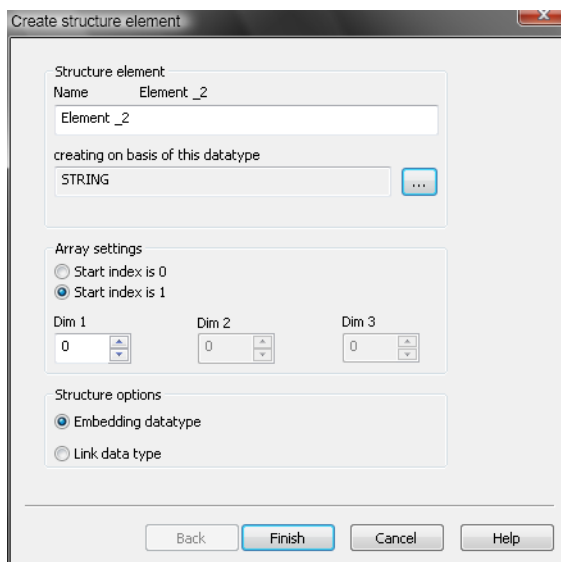
EMBED OR LINK

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!

6.2.3.2 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.

6.2.3.3 Changing the properties of a structure element

To change the properties of a structure element:

EMBEDDED STRUCTURE ELEMENT

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
    Note: If the renaming leads to an invalid variable name, a warning is shown and the variable is not activated. This also applies for XML import.

▶   Offsets

▶   Array dimensions

▶   Description

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

> **⚠Attention**
>
> 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!

6.2.3.4 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. In doing so, 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

6.2.3.5 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 deleted, all variables that are based on this structure data type are also deleted!

7 Drivers

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 real existing 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.

PROJECT MANAGER CONTEXT MENU

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:

Option	Description
Simulator driver SIMUL32	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	Variables of this driver are used for the calculation of mathematical functions or for operating hours counters.
System driver Sysdrv	variables for monitoring and controlling the hardware, the network and other project-specific properties.
Internal driver Internal	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.
Alternative Data Point driver ADP	

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 medium.

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

7.1 Driver detail view toolbar and context menu

CONTEXT MENU

The actions are only available if a screen has been selected in the detail list. Multiple selection is not possible for a driver.

Menu item	Description
New driver	Opens the dialog to create a new driver (on page 35)
Exchange driver	Opens the selection window to define a driver (on page 67).
Driver configuration	Opens the dialog for configuration of the selected driver.
Import variables from driver	Opens the dialog for online import of the variables from the PLC. **Note:** This action is only available for drivers that support online import.
Read PLC variables in background	Imports variables from the PLC and saves them in a file. **Note:** Only available if this is supported by the selected driver.
Cancel background reading	Cancels the current read process of PLC variables. **Note:** Only available if current PLC variables are read off.
Create/edit simulation project	Opens zenon Logic Workbench. ▸ If a driver simulation project has not yet been created, a new project is created in zenon Logic. ▸ If a simulation project has already been created, the existing simulation project is opened with the existing project configurations. ▸ For each driver, the attendant zenon Logic project is opened in a separate instance. **Info:** You can find the parameter settings for this in the driver's **Driver simulation project** property group.
Delete simulation project	Deletes the driver and the existing zenon Logic project for it. **Note:** Only available if a **Driver simulation project** has already been created for the selected driver.

Menu item	Description
Go back to original file	Jumps back to the initial position in the zenon Editor. **Note:** Only available in the context menu if a jump to the current position has been made from another position with the **Linked elements** context menu entry.
Delete	Deletes selected entries after a confirmation from list.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
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 cells that can be edited can be selected.
Replace text in selected column	Opens the dialog to search and replace texts for the selected column.
Properties	Opens the **Properties** window.
Help	Opens online help.

7.2 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.

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

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.

7.3 Creating a driver

In the **Create driver** dialog, you create a list of the new drivers that you want to create.
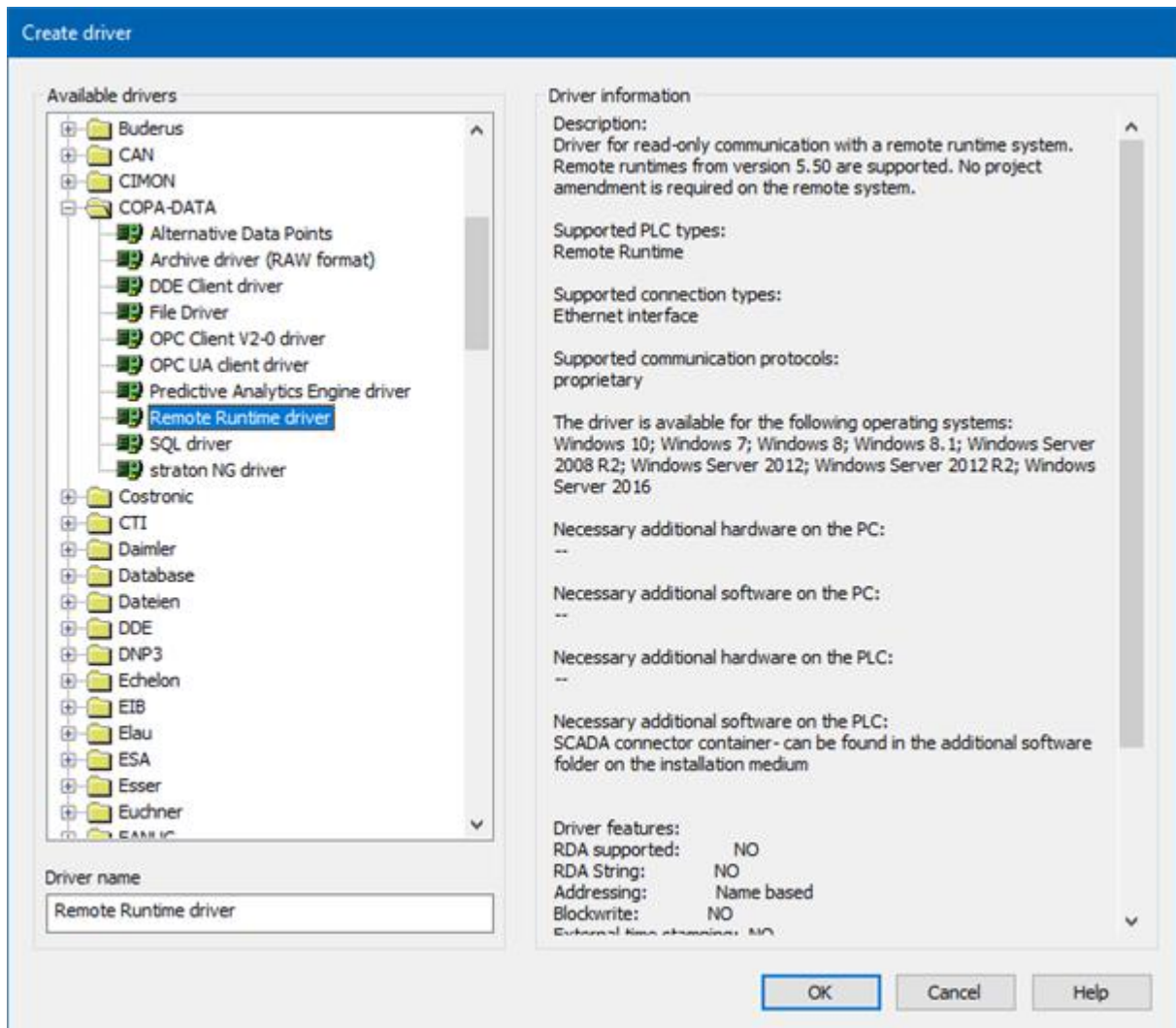
Parameter	Description
Available drivers	List of all available drivers.
	The display is in a tree structure:
	[+] expands the folder structure and shows the drivers contained therein.
	[-] reduces the folder structure
	Default: *No selection*
Driver name	Unique **Identification** of the driver.
	Default: *empty*
	The input field is pre-filled with the pre-defined

Parameter	Description
	Identification after selecting a driver from the list of available drivers.
Driver information	Further information on the selected driver. Default: *empty* The information on the selected driver is shown in this area after selecting a driver.

CLOSE DIALOG

Option	Description
OK	Accepts all settings and opens the driver configuration dialog of the selected driver.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

💡 **Information**

The content of this dialog is saved in the file called Treiber_[Language].xml. You can find this file in the following folder:
C:\ProgramData\COPA-DATA\zenon[version number].

CREATE NEW DRIVER

In order to create a new driver:

1. Right-click on **Driver** in the Project Manager and select **New driver** in the context menu.

   Optional: Select the **New driver** button from the toolbar of the detail view of the **Variables**.The Create driver dialog is opened.

   The **Create simple data type** dialog is opened.

2. The dialog offers a list of all available drivers.
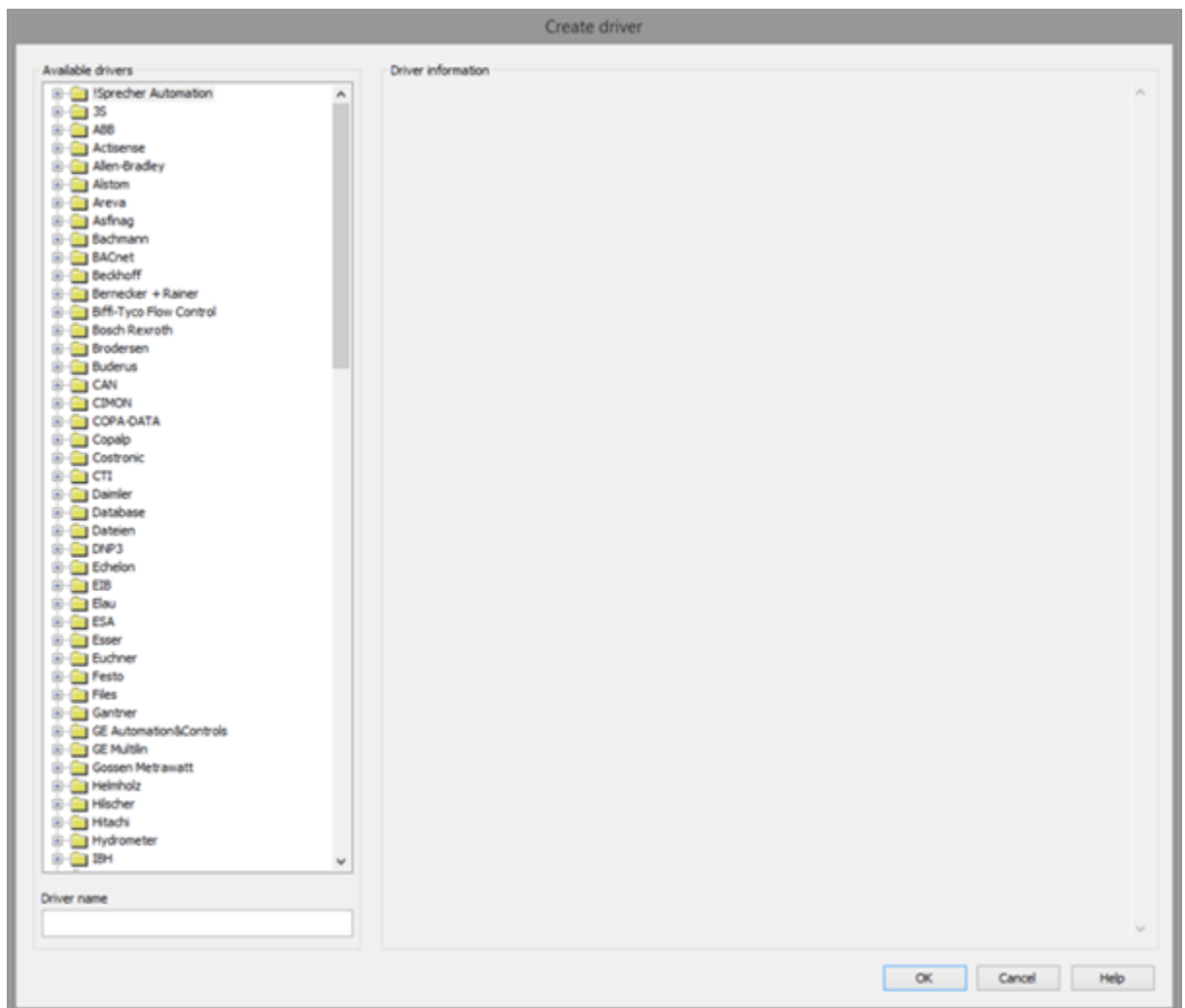
3. Select the desired driver and name it in the **Driver name** input field.
   This input field corresponds to the **Identification** property. The name of the selected driver is automatically inserted into this input field by default.
   The following is applicable for the **Driver name**:

   ▸ The **Driver name** must be unique.
   If a driver is used more than once in a project, a new name has to be given each time.
   This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.

   ▸ The **Driver name** is part of the file name.
   Therefore it may only contain characters which are supported by the operating system.
   Invalid characters are replaced by an underscore (_).

   ▸ **Attention:** This name cannot be changed later on.

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

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

DRIVER NAME DIALOG ALREADY EXISTS

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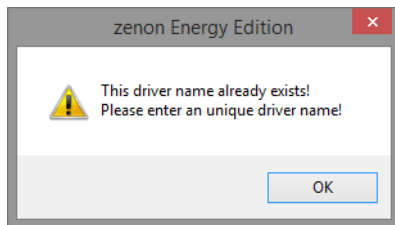
ZENON PROJECT

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

- ▸ **Intern**
- ▸ **MathDr32**
- ▸ **SysDrv**

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

7.4 Configuration of a driver

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

Option	Description
Mode	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*: No communication is established to the PLC. The

Option	Description
	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 (on page 44).
Keep update list in the memory	Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.
Output can be written	▸ *Active*: Outputs can be written. ▸ *Inactive*: Writing of outputs is prevented. **Note**: Not available for every driver.
Variable image remanent	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp. The variable image is saved in hardware mode if one of these statuses is active: ▸ User status *M1* (*0*) to *M8* (*7*) ▸ *REVISION(9)* ▸ *AUS(20)* ▸ *ERSATZWERT(27)* The variable image is always saved if: ▸ the variable is of the **Communication details** object type ▸ the driver runs in simulation mode. (not programmed simulation) The following states are not restored at the start of the

Option	Description
	Runtime:
	‣ *SELECT(8)*
	‣ *WR-ACK(40)*
	‣ *WR-SUC(41)*
	The mode **Simulation - programmed** at the driver start is not a criterion in order to restore the remanent variable image.
Stop on Standby Server	Setting for redundancy at drivers which allow only one communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade.
	Attention: If this option is active, the gapless archiving is no longer guaranteed.
	‣ *Active*: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status **switched off** but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.
	Default: *inactive*
	Note: Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.
Global Update time	Setting for the global update times in milliseconds:
	‣ *Active*: The set **Global update time** 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.
	Exceptions: Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the **Spontaneous driver update time**

Option	Description
	section.
Priority	The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.
	The variables are allocated separately in the settings of the variable properties. The communication of the individual variables can be graded according to importance or required topicality using the priority classes. Thus the communication load is distributed better.
	Attention: Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.

CLOSE DIALOG

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

UPDATE TIME FOR SPONTANEOUS DRIVERS

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

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

7.5   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 43): constant values simulated by the driver

▶   Simulation - counting (on page 43): values simulated by the driver are counted up

▶   simulation - programmed (on page 44): 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.

Note: **Simulation - programmed** is not supported by drivers for:

- Internal variables
- Mathematical variables
- Simulator variables
- System variables

> **💡 Information**
>
> zenon variables which represent the zenon Logic IO variables are not available in the project in state **Driver simulation programmed**.

7.5.1 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.

7.5.2 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.

> **💡 Information**
>
> With negative start values, the counting process only starts at 0.
>
> INT uses the maximum value of USINT for counting.
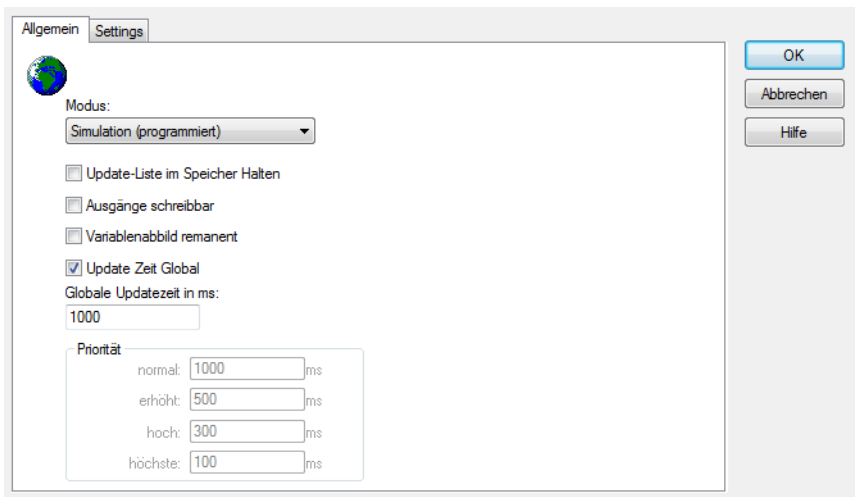
⚠️**Attention**

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.

7.5.3 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 50).

💡 **Information**

Windows CE

Simulation - programmed is not available for Windows CE.

7.5.3.1 Editor

7.5.3.1.1 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

> ## ⚆ Information
>
> Port numbers: At creating the simulation project for this project a distinct port number is assigned automatically. An automatically assigned port number is only unique for the respective project. In multi-hierarchical projects, it must be ensured that a port number is only used once in all projects that run in the Runtime. If a port number is used more than once, communication errors can occur with zenon Logic Workbench/Runtime.
>
> Port numbers can be amended manually. A port number must meet the following conditions:
>
> ▸ It must be free in the project.
>
> ▸ It must be unique for all projects that run in Runtime.
>
> ▸ It must be available on the computer on which the driver runs.
>
> Permitted port numbers:
>
> ▸ Minimum: *6000* (with automatic allocation, manually: recommended)
>
> ▸ Maximum: *6999* (with automatic allocation, manually: recommended)

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_<driver-ID>.zip** file in the *<Sql Projekt Pfad>**FILES*\zenon*custom**drivers* folder. 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.

> ### 💡 Information
>
> 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 TYPE

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* unsupported.
- ▶ 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 the 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 49) you must not create simulation projects
> offline. They cannot be deleted anymore.

7.5.3.1.2 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 49):**

  ▸ 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!

7.5.3.1.3 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 48) in *Offline* mode.
>
> A simulation project created *offline* in a **multi-user project** can therefore no longer be deleted.

7.5.3.1.4 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.

7.5.3.1.5 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.

7.5.3.2 Driver configuration

There are four modes available when configuring the driver:

▶ **Hardware**

- **Simulation - counting**
- **Simulation static**
- **Simulation - programmed**

Select the **simulation - programmed** mode.
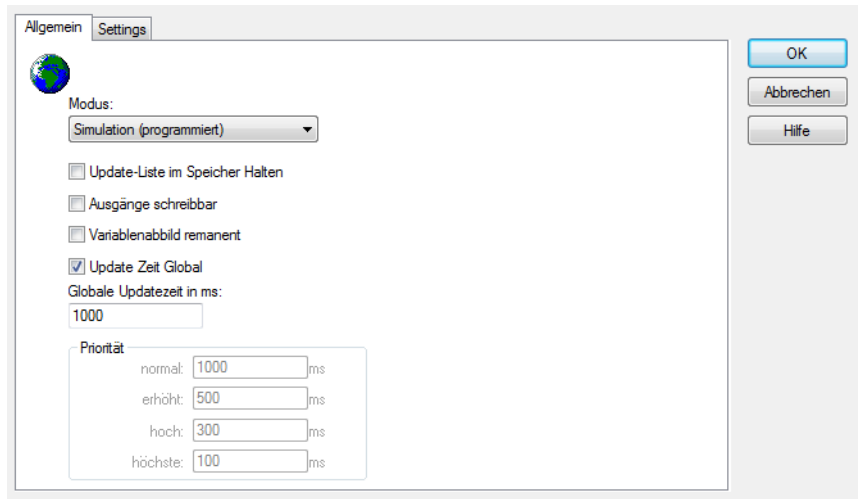
7.5.3.2.1 Driver variable status

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.

Bit	Meaning	Description
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 transmit 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.

💡 **Information**

The driver variable SimulRTState with Offset 60 displays the status of Runtime on the server. This is also true when the variable is displayed on the Standby.

7.5.3.2.2 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**
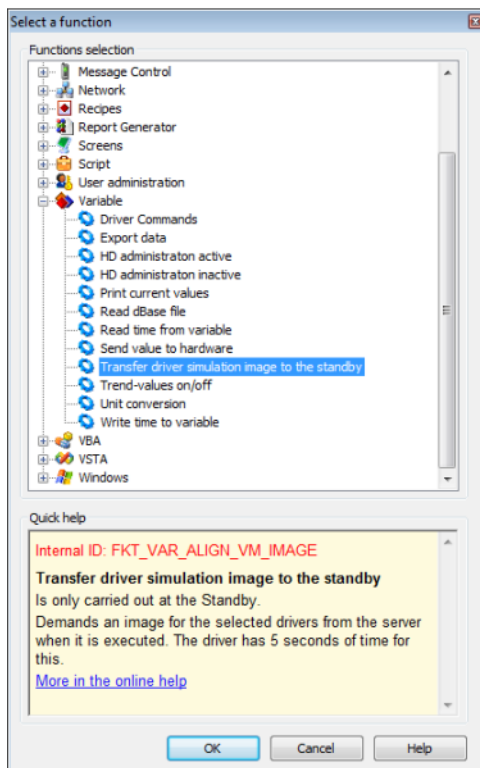
▶ The dialog to select the driver opens
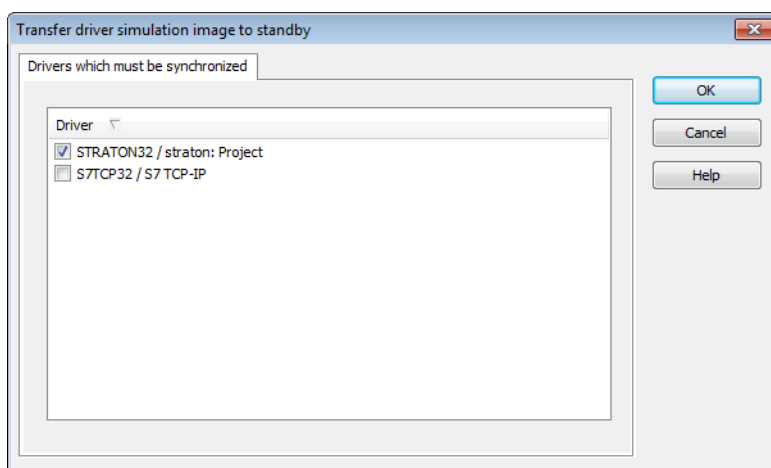
▶ In the list, all process-connected drivers (with the exception of the simulator) are available (clicking on the column heading changes the sorting)

▶ 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 Server** 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

7.5.3.3 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** and **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 45) 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: Execution mode	**Recommendation:** **Mode**: triggered **Cycle time**: 1/4 to 1/3 of the driver update time Too frequent value changes of simulated variables lead to increased computer utilization.
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 and reacted to in the simulation program.
"C" compiler	Native code is supported. Name of the DLL that the compiled code must contain: **t5app.dll**.

7.5.3.4 Runtime

7.5.3.4.1 Functionality

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 57))

▶ 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 in the Workbench and in the Diagnosis Server

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

▶ Online Change

▶ Programming languages:

  ▶ *SFC* - Sequential Function Chart

  ▶ *FBD* - Function Block Diagram

  ▶ *LD* - Ladder Diagram

  ▶ *ST* - Structured Text

- ▸ *IL* - Instruction List - obsolete
- ▸ 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

7.5.3.4.2 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 additional 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.

7.5.3.4.3 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 subfolder and all information contained in it is transferred to the target computer

7.5.3.4.4 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 the Runtime.

TRANSFER DRIVER SIMULATION IMAGE TO STANDBY FUNCTION

This function is only executed on the Standby Server (for configuration, see Cyclical comparison (on page 52) 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 in the 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 the Runtime.

DRIVER WRITE COMMAND

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

Write operations are carried out by the driver asynchronously in the 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 executed after their values have been changed in the 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.

7.5.3.4.5 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 39).

7.5.3.4.6 Redundancy

START ON THE STANDBY SERVER

If the driver is already running on a Standby Server, 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 52) 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.

SYNCHRONIZATION

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**.

7.5.3.4.7 Status

STATUS SIMULATION

The **Create status bits for variables with profile** option must be active (*yes*) for status simulation. You can find this option in zenon Logic in the **Options** dialog. To call up this dialog, select the **Parameter** entry in the **Project** menu bar. You can find this option in the **Compiler** options group. Result: 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)

> 💡 **Information**
>
> 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 zenon 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 *Spontaneous* is set. This is the state set if status handling is turned off or no status was defined. State *GI + INAVLID* is deleted.
18: Invalid (INVALID/_VSB_I_BIT)	*INVALID* status is set. *SPONTAN + GI*status is deleted.
16: General query (GI/_VSB_GR)	Status *General query* is set. State *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 status 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.

7.5.3.4.8 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.

> ### ⚙ Information
>
> 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.

7.5.3.4.9 Driver commands

Driver commands (on page 201) are used to influence drivers using zenon; start and stop for example. 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*

7.5.3.4.10 Variable assignment

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*)*. **Driver variable** driver-object-type variables are never requested by the simulation.

7.5.3.4.11 Timestamp

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 exception is the IEC870 driver however; this also transfers new timestamps from the hardware (or **simulation - programmed**) with the same value and same status to the Runtime.

> ☀ **Information**
>
> 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.

7.5.3.5 Notes on 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 yet have valid addressing for 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. **Rename**

   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. **Integrated solutions**

   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. **Communication based on variable names**

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

5. **Deleting or importing variables**

   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. **Variables with active "**_embedded_**" symbol**

   Variables which exist in a program and whose "_embedded_" symbol is active are not available via the **Logic to SCADA** connection in zenon. If a variable with an active "_embedded_" symbol is moved from the global area to a program, the variable is deleted in zenon.

7.5.3.6 Error message

Engineering in the zenon Editor

Error message	Cause and solution
The selected name already exists for another driver or contains invalid characters! Enter a different 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 the 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 Hint: 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.

7.6 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 **Replace driver** command.
   The same dialog (on page 35) as when creating a driver is opened.

4. 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).

7.7   Delete Driver

To delete drivers:

1. Select the desired driver (multiple selection is possible).

2. Select **Delete** in the context menu or in the toolbar.

3. The drivers are deleted

    The deletion process must be confirmed with a confirmation request.

Attention: All variables belonging to it are also deleted.

7.8   Communication details

The driver kit implements a number of driver variables. This variables are part of the driver object type *Communication details*. These are divided into:

> ▸   Information

> ▸   Configuration

> ▸   Statistics and

> ▸   Error message

The definitions of the variables implemented in the driver kit are available in the import file **DRVVAR.DBF** and can be imported from there.
Path to file: *%ProgramData%\COPA-DATA\zenon<Versionsnummer>\PredefinedVariables*

Note: Variable names must be unique in zenon. If driver variables of the driver object type *Communication details* are to be imported from **DRVVAR.DBF** again, the variables that were imported beforehand must be renamed.

💡 **Information**

Not every driver supports all driver variables of the driver object type *Communication details*.

For example:

▸ Variables for modem information are only supported by modem-compatible drivers.

▸ Driver variables for the polling cycle are only available for pure polling drivers.

▸ Connection-related information such as **ErrorMSG** is only supported 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

Name from import	Type	Offset	Description
LineStateSpecialInfo	*BOOL*	24.7	Special status information received
LineStateConnected	*BOOL*	24.8	Connection established
LineStateProceeding	*BOOL*	24.9	Dialing completed
LineStateOnHold	*BOOL*	24.10	Connection in hold
LineStateConferenced	*BOOL*	24.11	Connection in conference mode.
LineStateOnHoldPendConf	*BOOL*	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	*BOOL*	24.13	Connection in hold for transfer
LineStateDisconnected	*BOOL*	24.14	Connection terminated.
LineStateUnknow	*BOOL*	24.15	Connection status unknown
ModemStatus	*UDINT*	24	Current modem status
TreiberStop	*BOOL*	28	Driver stopped For *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 state of Runtime for driver simulation.
ConnectionStates	*STRING*	61	Internal connection status of the driver to the PLC. Connection statuses: ▸ *0:* Connection OK ▸ *1:* Connection failure ▸ *2:* Connection simulated Formating: **<Net address>:<Connection status>;...;...;** A connection is only known after a variable has first signed in. In order for a connection to be contained in a string, a variable of this connection must be signed in once.

Name from import	Type	Offset	Description
			The status of a connection is only updated if a variable of the connection is signed in. Otherwise there is no communication with the corresponding controller.

CONFIGURATION

Name from import	Type	Offset	Description
ReconnectInRead	BOOL	27	If TRUE, the modem is automatically reconnected for reading
ApplyCom	BOOL	36	Apply changes in the settings of the serial interface. Writing to this variable immediately results in the method SrvDrvVarApplyCom being called (which currently has no further function).
ApplyModem	BOOL	37	Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings **PhoneNumberSet** and **ModemHwAdrSet**.
PhoneNumberSet	STRING	38	Telephone number, that should be used
ModemHwAdrSet	DINT	39	Hardware address for the telephone number
GlobalUpdate	UDINT	3	Update time in milliseconds (ms).
BGlobalUpdaten	BOOL	4	TRUE, if update time is global
TreiberSimul	BOOL	5	TRUE, if driver in sin simulation mode
TreiberProzab	BOOL	6	TRUE, if the variables update list should be kept in the memory
ModemActive	BOOL	7	TRUE, if the modem is active for the driver
Device	STRING	8	Name of the serial interface or name of the modem
ComPort	UINT	9	Number of the serial interface.
Baudrate	UDINT	10	Baud rate of the serial interface.

Name from import	Type	Offset	Description
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.

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

ERROR MESSAGE

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

Name from import	Type	Offset	Description
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

7.9 Driver documentation

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

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

8 Create, modify and use variables

Variables can be created:

▸ as simple variables (on page 78)

▸ in arrays (on page 83)

▸ as structure variables (on page 94)

VARIABLE DIALOG

To create a new variable, regardless of which type:
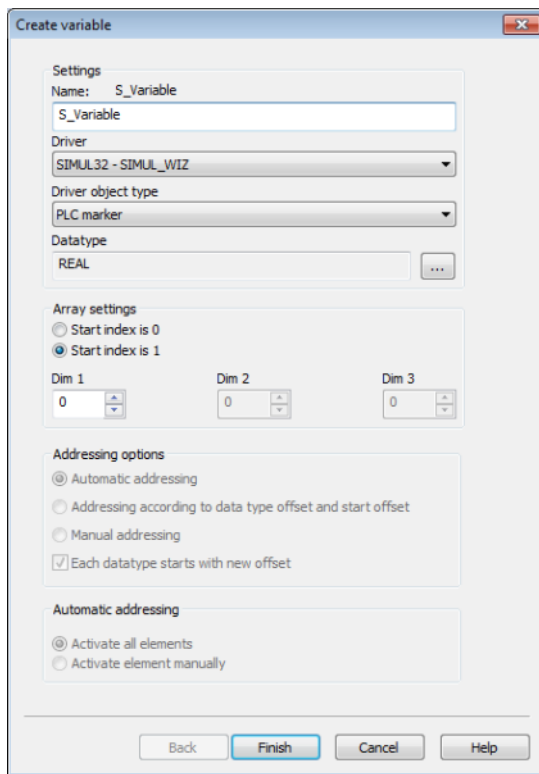
1. Select the **New variable** command in the **Variables** node in the context menu
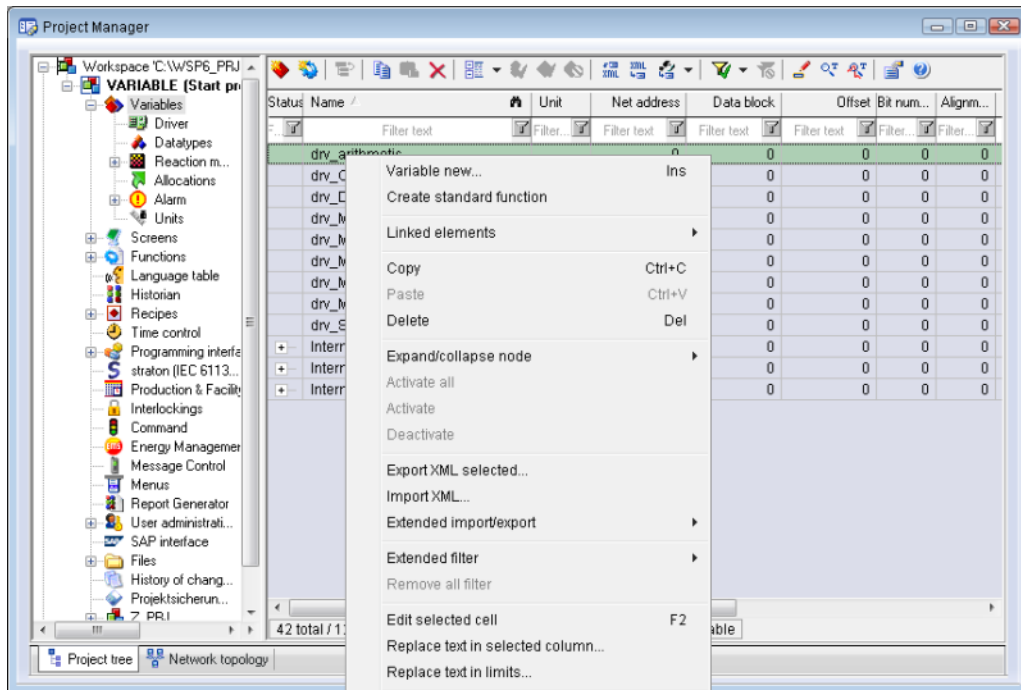
The dialog for configuring variables is opened

2. Configure the variable

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

CREATE VARIABLE 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 this name.
	Maximum length: 128 characters
	Attention: the characters **#** and **@** are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive. **Note:** Some drivers also allow addressing using the **Symbolic address** property.
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**) is automatically loaded.
Driver Object Type	Select the appropriate driver object type from the drop-down list.
Data Type (on page 15)	Select the desired data type (on page 15). Click on the ... button to open the selection dialog.
Array settings	Expanded settings for array variables. You can find details in the

Property	Description
	Arrays (on page 83) chapter.
Addressing options	Expanded settings for arrays (on page 83) and structure variables (on page 94). You can find details in the respective section.
Automatic element activation	Expanded settings for arrays (on page 83) and structure variables (on page 94). You can find details in the respective section.

SYMBOLIC ADDRESS

The **Symbolic address** property can be used for addressing as an alternative to the **Name** or **Identification** of the variables. Selection is made in the driver dialog; configuration is carried out in the variable property. When importing variables of supported drivers, the property is entered automatically.

Maximum length: *1024* characters.

The following drivers support the **Symbolic address**:

- **3S_V3**
- **AzureDrv**
- **BACnetNG**
- **IEC850**
- **KabaDPServer**
- **OPCUA32**
- **Phoenix32**
- **POZYTON**
- **RemoteRT**
- **S7TIA**
- **SEL**
- **SnmpNg32**
- **PA_Drv**
- **EUROMAP63**

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.
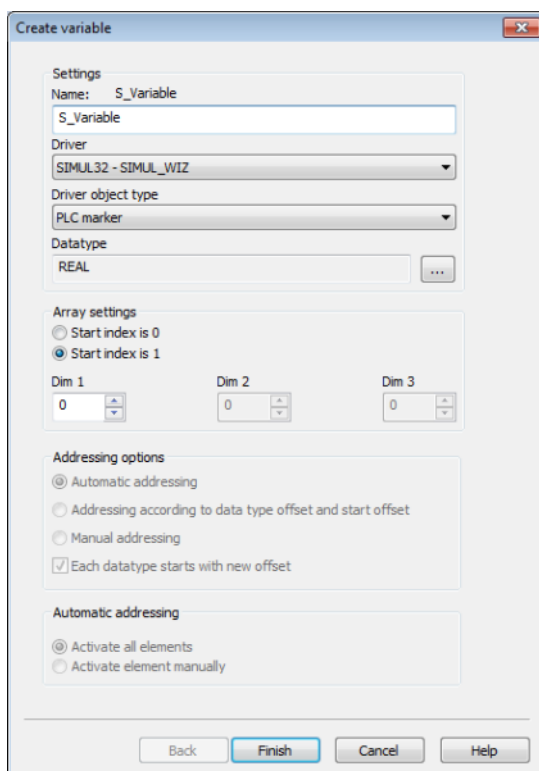
8.1 Simple variables

Simple variables always consist of a data type (on page 15) and a driver object type. They form the basis for array variables (on page 83) and structure variables (on page 94).

8.1.1 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 34) and data type (on page 15) 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.
Drivers	Select the desired driver from the drop-down list.

Property	Description
Driver Object Type	Select the appropriate driver object type from the drop-down list.
Data Type	Select the desired data type (on page 15). 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.

8.1.2 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.

8.1.2.1 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. **Drivers**
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.
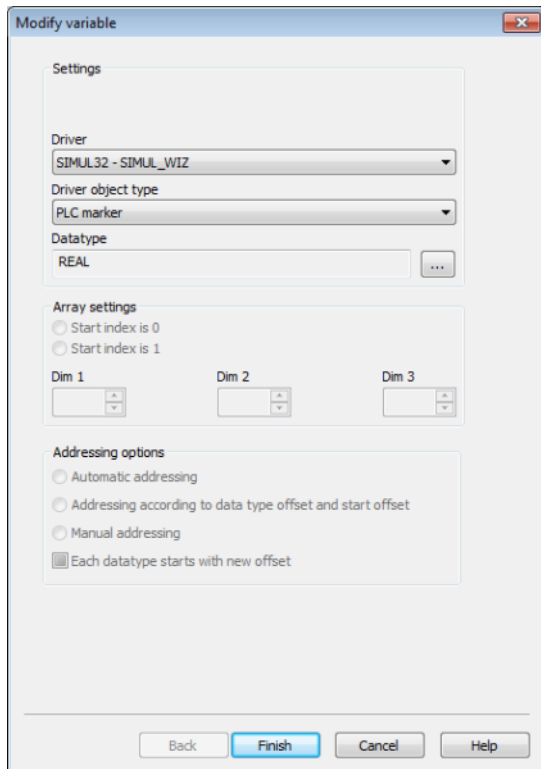
MODIFICATION, IF SEVERAL VARIABLES HAVE BEEN SELECTED:

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 multiselect of variables, the checkboxes have 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 81)', after the modification in the output window of the Editor.

8.1.2.2 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.
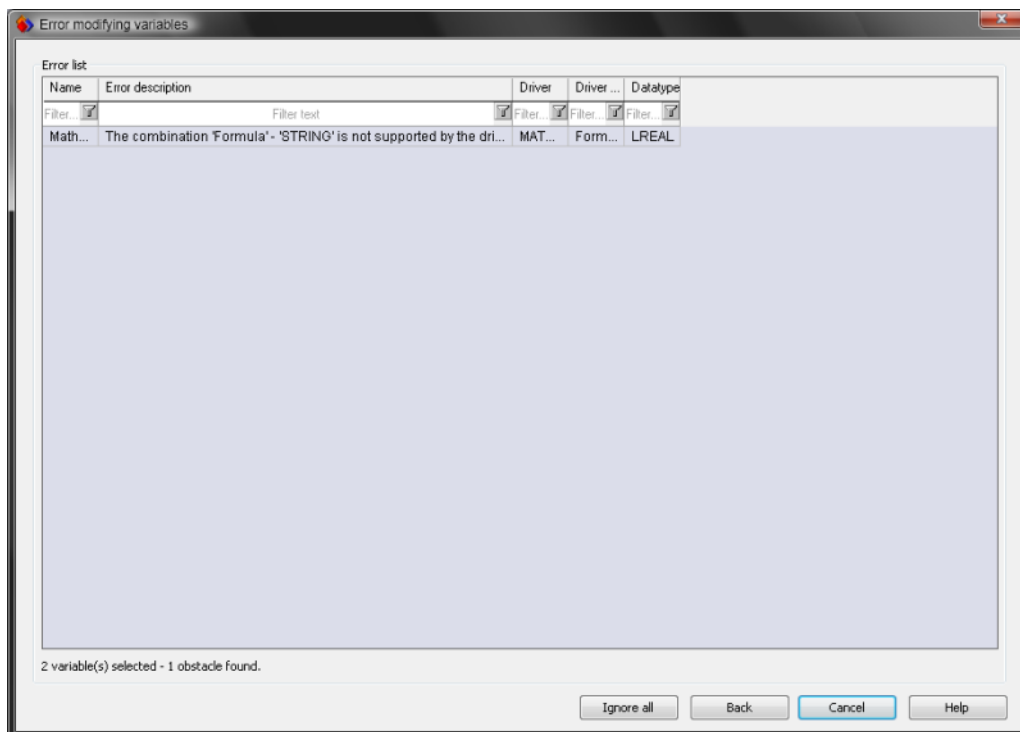
Parameter	Description
Name	Name of the variable, which cannot be changed.
Driver	Name of the driver with the driver identification, on which the variable is based.
Data type	Datatype used by the variable.

Parameter	Description
Driver Object Type	Driver object type that uses the variable.
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.

8.1.3 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:
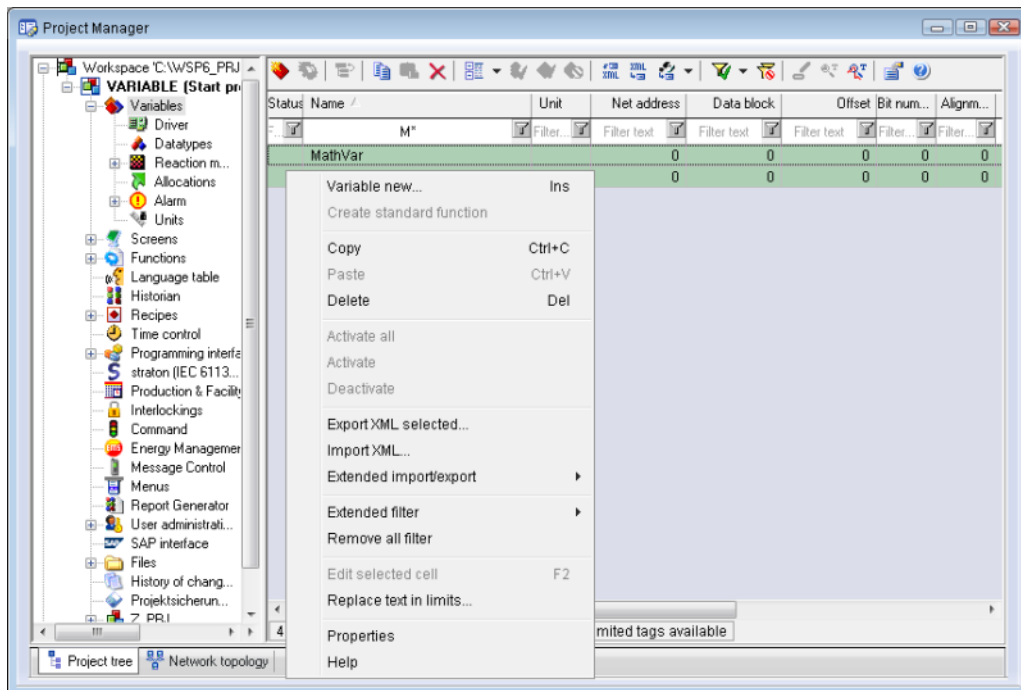
Clicking on **Delete** opens the following security dialog:
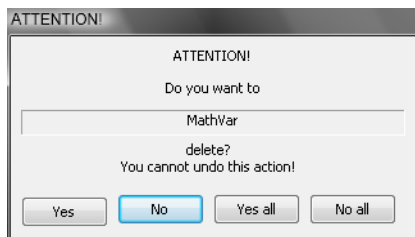
8.2 Arrays

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 94). 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.

BLOCK ARRAYS

With block arrays, many values are read and written at once and can be displayed in Extended Trend, for example. You can find detailed information in the Block Arrays (on page 84) chapter.

8.2.1 Block arrays

Block arrays are variables that contain a complete block (memory area) of a controller.

The **Block array size** property in the **additional settings** property group can be used to set the size of this block.

The block size depends on the driver used as well as the driver's object type.

Example: With the **S7TCP32 driver** and the *extended datablock* object type, a block with size *10* has a memory range of *10* bytes. With a **Modbus RTU driver** and the *Holding Register* object type, the block with size *10* has a memory area of *20* bytes.

Block arrays need less system resources than individual variables and make the following possible at once:

▸ The reading and writing of many values from the PLC by means of a driver (with the exception of internal drivers).

▸ The display of many values in Extended Trend.
  In Extended Trend, **block array** can be given as the data origin type. All values of the array are then shown in a curve.

▸ The processing of many values as recipe values in the RGM.
  Each block array is shown individually in the RGM: <Variable name>[Index of 0-n], for example: BlockArrayVariable[0] - BlockArrayVariable[500]

▸ The further processing of many variables via the API.

A further application is the transfer of external values to zenon via the API.

In the **Recipegroup Manager**, the elements of the block array are displayed with the suffix [*0*] to [*n*].

Only one tag per block array is needed for licensing. Only one time stamp and status is used per block array.

With the numerical drivers, the offset, which is set on the basis of the variable, is the start offset. The values are read from this offset.

With symbolically-addressed drivers, a block with the set size is read and/or written from the given symbol. This is not supported by every driver however, and it also depends on the communication protocol and the controller addressed.
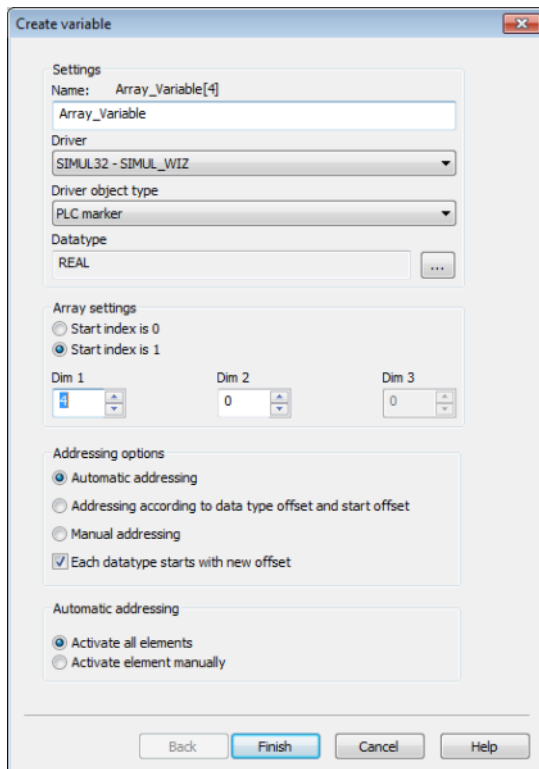
The addressing of block arrays, whereby the data types are smaller than the driver granularity, is also not supported in all cases. This means that each new element of the block array gets an offset that has been increased by one. Depending on the driver, this is a **byte**, a **word** or a **double word**.

Attention: No other variables with the same name can exist in the project!

Note: The use of block arrays is not supported by all drivers. It is thus possible, when using the **internal driver** in the Editor, for a value to be defined but with no variables displayed in the Runtime (value is always *0*).

8.2.2 Create array variable

Simple array variables are created like simple variables (on page 78).

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.
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) 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 15). Click on the ... button to open the selection dialog.
Automatic addressing	zenon calculates an automatic (on page 87) 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.

Property	Description
Manual addressing	Manual (on page 91) addressing.
Each data type 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.

8.2.3 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

▶ Drivers

▶ Array dimension

8.2.3.1 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: arbitrary entry

- ▶ 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 **Finish**, the five defined variables are created in the variable list. In addition, 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.

8.2.3.2 Example for an array with automatic addressing for a BOOL variable

We proceed as in the example of automatic addressing (on page 88) 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 BOOL[1]	0	50	100	0
Array of BOOL[2]	0	50	101	0
Array of BOOL[3]	0	50	102	0
Array of BOOL[4]	0	50	103	0
Array of BOOL[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 BOOL[1]	0	50	100	0
Array of BOOL[2]	0	50	100	1
Array of BOOL[3]	0	50	100	2
Array of	0	50	100	3

Array	Net address	Data block	Offset	Bit number
BOOL[4]				
Array of BOOL[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.

8.2.3.3 Example of an array with manual addressing:

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

▶ Net address

▶ Offset

▶ Data block (optional)

▶ Bit number (optional)

This could look as follows for the example of an array with automatic addressing for a BOOL variable (on page 90):

Array	Net address	Data block	Offset	Bit number
Array of BOOL[1]	0	50	100	0
Array of BOOL[2]	0	50	100	2
Array of BOOL[3]	0	50	101	0
Array of BOOL[4]	0	50	101	2
Array of BOOL[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 recalculated.
>
> 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 variable is thus retained.

8.2.4 Changing the properties of an array
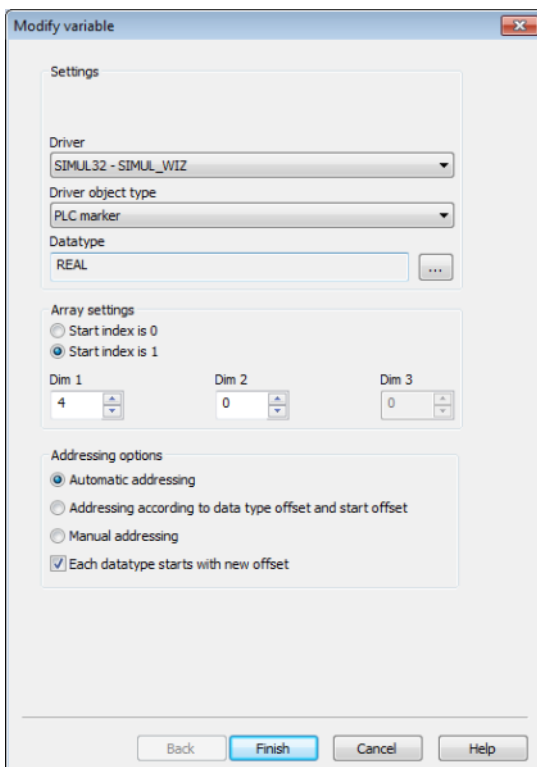
The properties are changed using the Properties window and the **Modify variable** dialog.

Note two special properties in particular:

▶ **last used offset:** Shows which offset (including bit number) was the last used in automatic addressing.

▶ **Size in byte:** 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) 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 15). Click on the … button to open the selection dialog.
Automatic addressing	zenon calculates an automatic (on page 100) 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 103) 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:

▶ **Signal range min.** only adapted if SignalbereichMinNeu < SignalbereichMinAlt

▶ **Signal range max.** only adapted if SignalbereichMaxnNeu < SignalbereichMaxAlt

8.2.5 De/activating array elements

Each array element can be activated or deactivated 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.

8.3 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 15)/Structure data types (on page 26) 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 |

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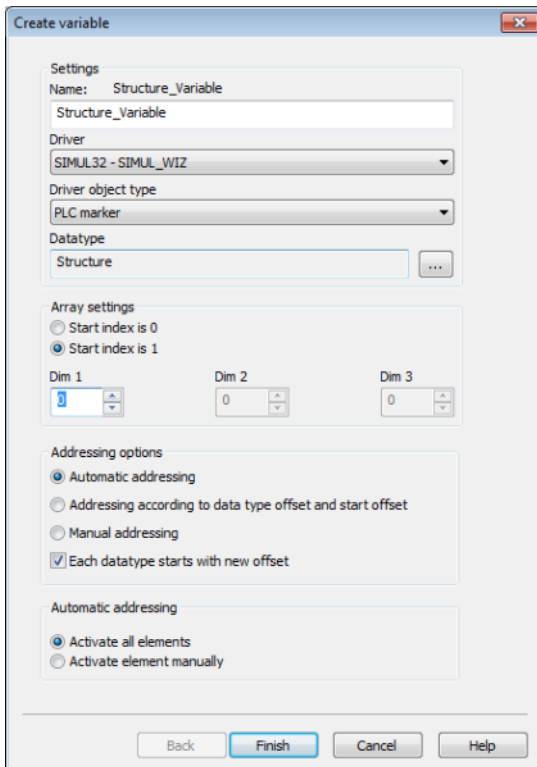
At the import of structure variables, active and inactive variables are imported. Existing imports are not overwritten at the import. If an inactive variable is imported to a project and then activated, it stays active even after a new XML import.

8.3.1 Changing structure variables

To create a structure variable:

1. Create a simple variable (on page 74)

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.
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) 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 15). Click on the … button to open the selection dialog.
Automatic addressing	zenon calculates an automatic (on page 100) 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.

Property	Description
Manual addressing	Manual (on page 103) 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*

8.3.1.1 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 95) 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 data type 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

Variable	Net address	Data block	Offset	Bit number
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.

8.3.1.2 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 97), 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:

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.On/Off[1]	0	50	103	0
Engine Speed Control[1].Engine Speed.On/Off[[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.On/Off[[2]	0	50	108	1

Variable	Net address	Data block	Offset	Bit number
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.On/Off[[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.On/Off[[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.

8.3.1.3 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 97), 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.
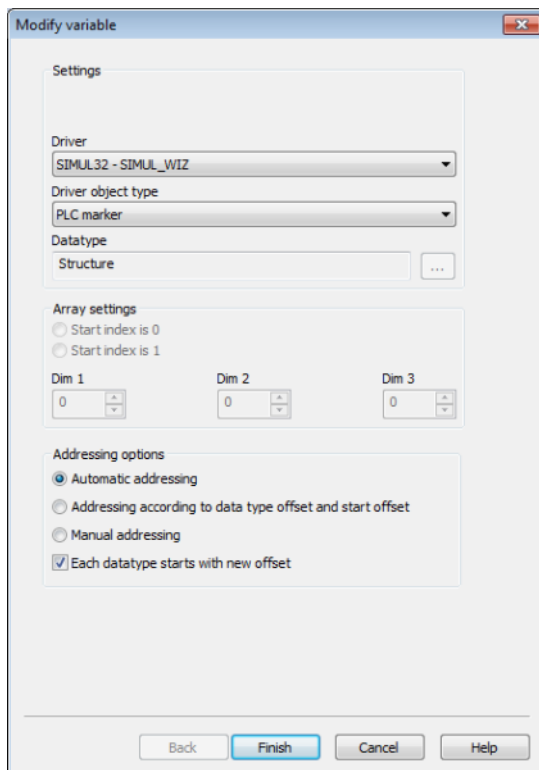
If the start-offset of a structure variable is changed, the offsets of the variables already activated are also changed by the delta of the change. The offset difference between the activated variable and the array variables is thus retained.

8.3.2 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) is automatically loaded.
Driver object type	Select the appropriate driver object type from the drop-down

Property	Description
	list.
Data type	Select the desired data type (on page 15). Click on the … button to open the selection dialog.
Automatic addressing	zenon calculates an automatic (on page 100) 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 103) 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).

8.3.2.1 Changing the properties

Changes are made in the properties window.

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 110) chapter in relation to this)

8.3.2.2 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.

8.3.2.3 Changing the sequence

A change to the sequence of the structure elements of a structure variable can only be made in the corresponding structure data type.

When the structure data type is changed, all structure variables based on it are also changed.

8.3.3 Deleting structure variables

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

8.4 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 the Runtime:

▶ Be loaded

▶ Contactable in the network zenon does not check the contactability in the network in the 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**
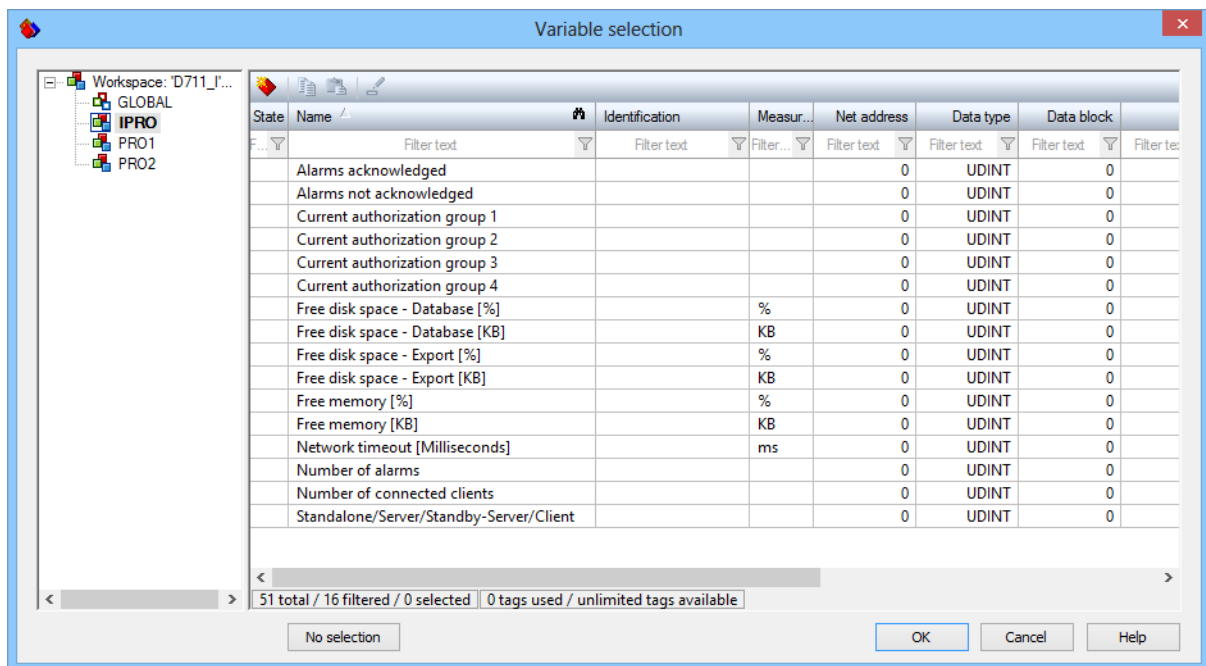
If cross-project use is not possible, projects in the variable selection dialog are displayed as grayed out.

Note: The size of this dialog can be adjusted. The position and size of the dialog window are saved in the Editor, Runtime and zenon Web Client independently of the project.

> **⚠Attention**
>
> 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.
>
> Note the special conditions for cross-project variables in the command processing.

8.5   Use in zenon Analyzer

Variables can be provided with specific reports information for the use in COPA-DATA product zenon Analyzer. There is a separate group of properties available for this.

The following properties in the zenon **Analyzer** variable properties group provide information for reports in the zenon Analyzer:

▶ **Visual name**: Entry of a display name of the variable in zenon Analyzer. This must be unique in the project. The check is not carried out when issued in zenon, but when imported into zenon Analyzer. If this property is changed after the first export to a zenon Analyzer, these changes are not applied in the zenon Analyzer.

▶ **Meaning**: Entry of the (Meaning) of a variable in the zenon Analyzer. Entry is manual or by means of the **Meaning and Waterfall Chart Wizard**. Several meanings are separated by a comma.
Syntax:*[Meaning1],[Meaning2],...,[MeaningN]*

▶ **Parameters for waterfall diagram**: Parameters of a variable for a waterfall diagram in zenon Analyzer. Entry is manual or by means of the **Meaning and Waterfall Chart Wizard**. The individual parameters are separated by a comma. Several waterfalls are divided by a semicolon.
Syntax: *[model name],[row index],[index in row],[color code];*

The data takeover from zenon into zenon Analyzer is possible by an export with the **Analyzer Export Wizard**.

8.5.1 Export of waterfall diagrams

A waterfall diagram can be used for either **line-based reports** or for **machine-based reports**. The parameters for the diagram are stored in the **Analyzer**/**Parameters for waterfall diagram** variable property from zenon 7.20. These can be entered manually or created with the **Meaning and Waterfall Chart Wizard**.

EXPORT

The wizard reads the **Parameters for waterfall diagram** property when loading the zenon workspace. If there are correct entries, these these are exported in the background and written to the database of the zenon Analyzer.

STRUCTURE OF THE ENTRIES

Depending on the structure of the entries, a decision is made on whether it is entries for machine-based or line-based diagrams.

▶ Machine based: Structure with 4 digits, separated by a comma; ended with a semicolon.

Syntax: **[model name],[line index],[column index],[color code code];**
Example: **MyWaterfall,4,2,#80FF00;**

▶ Line-based: Structure with 7 digits, separated by a comma; ended with a semicolon.

Syntax: **[model name],[line index],[column index],[color code],[loss of auxiliary machine],[add loss of auxiliary machine],[subtract loss of auxiliary machine];**
Example: **MyLineAnlaysis,4,2, #80FF00,0,0,0;**

RULES FOR READING:

The following is applicable for reading:

▸ If there are entries for **Parameters for waterfall diagram**, corresponding entries in the **Resources label** field are ignored.
The structure decides whether the entry can be evaluated as machine-based or line-based.

▸ The identification **WF=** is not necessary but can to be used. The individual elements of a model are separated by a comma. If several waterfall models are assigned to a variable, a semicolon is used as a separator.

▸ If there are no entries, corresponding entries from the **Resources label** are accepted. The identification **WF=** must be prefixed here.

▸ With versions of zenon before 7.20, the waterfall parameters are taken from the zenon **Resources label** property.

8.6 Linking variables/data type with style group/scale style/curve style

The **Extended Trend** node is now available in the properties of the variables and the data type.
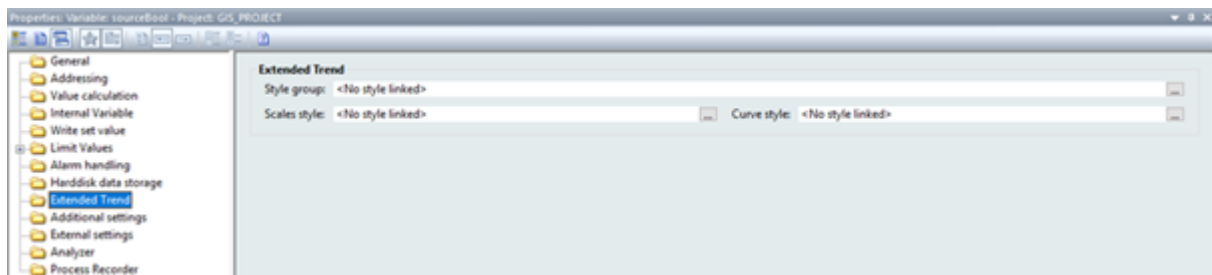
There you can assign predefined styles or a predefined style group to a variable or a data type.

The following selection possibilities are available:

Property	Description
Style group	Allows the selection of a pre-defined style group.
Scales style	Allows the selection of a pre-defined scale style.
Curve style	Allows the selection of a pre-defined curve style.

Engineering:

1. Click on the respective **…** selection button.

2. Select the desired style group or the desired style.

3. Confirm the selection by clicking on **OK**.

The selected styles are used in the Runtime for the display of the Extended Trend.

9 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 possibility 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 is changed in the base object, all objects that are derived from this base object also accept the change. The individual objects are not saved with the derived object directly, but are always referenced by the base object.

Exception: As soon as a value for a derived object is overwritten (changed), this property loses the reference to the base object (only for the inheritance). If such an overwritten property is changed in the basic object, these changed no longer effect the inherited object. However this only concerns the property that was interrupted in the connection to the base object. All other properties are applied by the basic object as before. An overwritten property can be turned back into a derived property at any time.

9.1 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.

9.1.1 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

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 something is changed in this data type, these changes directly affect the variables **Engine.Speed Control.Speed Actual**.

So be careful with changes in data types.

Properties are overwritten or the derivation from the data type is carried out as described further above.

9.2 Inheriting properties of a datatype with simple variables

All properties are transferred from a data type when a new variable is created. This is visualized for the corresponding properties with a symbol (a bent arrow on a frame).

INHERITED PROPERTIES ARE MARKED WITH A CHECKMARK

In this example, all properties are inherited from the data type except the *Unit* property. 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**
>
> Changes to the data type always have an effect on all variables that are based on this data type. A sole exception to this are variable properties for which the values taken from the data type have been amended.

9.2.1 Overwriting properties

Properties of variables can be overwritten in two ways:

▸ The property is changed using keyboard or mouse input. As soon as the change is applied, this connection's property is separated from the data type.

▸ The property is separated from the data type using the corresponding context menu. In doing so, either the selected property can be selected from the data type or all properties of the variable. (screenshot!)

the tick disappears as soon as a property is overwritten. All changes to a data type then no longer have an effect on these overwritten properties of a variable.

9.2.2 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.

10 Value calculation

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

> **⚠Attention**
>
> When using variables in zenon Logic and in the combined element, the value amendment is not taken into account. It is always used by the raw value obtained from the PLC.

HYSTERESIS AND THRESHOLD VALUE

Quickly-changing (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 quickly-changing values from the driver to variables.

▶ Hysteresis for archive: Prevents the creation of archive entries for quickly-changing values in relation to the measurement range.

▶ **Threshold value**: Prevents unwanted reactions such as alarms for quickly-changing numerical values (on page 119).

LINEAR VALUE ADJUSTMENT

Settings for linear conversion of the PLC values to visualization value.

The **Signal range min.** and **Signal range max.** properties, as well as **Measuring range min.** and **Measuring range max.** determine the offset and increase of the setting line. These are applied to all received values. This is also applicable to values outside of the signal range.

Properties:

▶ **Signal range min.**: Minimum value of the variables in the PLC.

▶ **Signal range max.**: Maximum value of the variables in the PLC.

▶ **Measuring range min.**: Minimum value of the variables in zenon.

▶ **Measuring range max.**: Maximum value of the variables in zenon.

NON-LINEAR VALUE AMENDMENT

Variables whose raw value (controller) cannot be scaled in a linear manner in the measured value (zenon) can be amended using the **Non linear value adjustment with macros** property. In doing so, raw values are amended using VBA macros in the PLC and by means of a macro in zenon. The required macros must already be available in the zenon **macro list**.
They are selected using the following properties:

▶ **AdjustHardwareValue**: Is called up in Runtime each time there is a change to the raw value in the PLC and amends the value in zenon.

▶ **AdjustScadaValue**: Is called up when writing a value from zenon and amends the raw value on the PLC.

> ⚠ **Attention**
>
> Both properties must always be configured. The **AdjustHardwareValue** macro must be created inversely to **AdjustScadaValue** as an inverse function.

10.1 Hysteresis

Hysteresis defines the area within which a value change is ignored. This prevents quickly-changing (e.g. fluctuating) values that would constantly trigger limit value violations 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 and **Negative for signal** properties 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 and **Negative for archive** properties for measuring range

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

▶ **Zero-point suppression**
Values that change that are within hysteresis are not transferred from the driver to variables. This evaluation takes place before hysteresis.

If the project configuration contains the value *0*, the zero clamping is not taken into account. All values are transferred from the driver to the variable and displayed in the Runtime.

> ## 🖹 Example
>
> A limit value of 50° C is defined. The temperature normally fluctuates somewhat. Therefore the limit value is violated each time the 50° limit is exceeded or gone under, which triggers an alarm and an entry in the archive.
>
> To prevent this, the following is defined as hysteresis for the variable:
>
> ▸ **Positive for signal**: *2*
>
> ▸ **Negative for signal**: *2*
>
> And for the archive as hysteresis:
>
> ▸ **Positive for archive**: *8*
>
> ▸ **Negative for archive**: *6*
>
> Zero clamping is also configured:
>
> ▸ **Zero-point suppression**: *20*
>
> Thus:
>
> ▸ An actual value of *-20°* to *+20°* is always displayed with the value *0* in the Runtime.
>
> ▸ The value of a variable for an actual value of *-20°* to *+20°* is set to the value *0*.
>
> ▸ Only once *52°* has been exceeded or   *48°* has been gone below is this considered a limit value violation
>
> ▸ An archive entry is only created once *58°* has been exceeded and *44°* has been gone below.
> Because a **Zero-point suppression** has been configured, the archive entry is only created for the temperature range from *20°* to *44°*.

Note: Hysteresis that is to avoid unwanted reactions (such as alarms) to limit value violations due to quickly-changing (fluctuating) values can be created using threshold values (on page 119).

> ## 💡 Information
>
> Note:
>
> ▸ Not every driver supports hysteresis. You can find information on whether your driver supports hysteresis in the documentation for the corresponding driver.
>
> ▸ Integration project:
> If hysteresis is configured for a variable in a subproject, this has no effect if the variable is in an archive of the integration project.

11 Limit Values

Limit values have the task to trigger a reaction, as soon as a limit value 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

  ▶ Limit values 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 value**" 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. Reaction matrices can also be used for string values. More information on reaction matrices can be found in the chapter Reaction matrix (on page 131).

Note: Limits are not deleted automatically as they may be lost for a wrong variable type selection. For exampel if you change a numeric variable with a defined limit to type *BOOL*, this variable will have three limits.

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 **Limit Values** group

    Note: xxxxxx is the placeholder for the parameters to be transferred

  ▶ 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.

11.1 Defining limit values 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 value, click **New limit**. Each time you click on **New limit value**, a new limit value (limit value 1, limit value 2, ...) is added to the list below.

Note: Not available if a reaction matrix has been linked via the **Reaction matrix** property.

All available properties for defining limit values are listed in the properties window and are described in the properties help. The context sensitive help is automatically displayed when selecting a property.
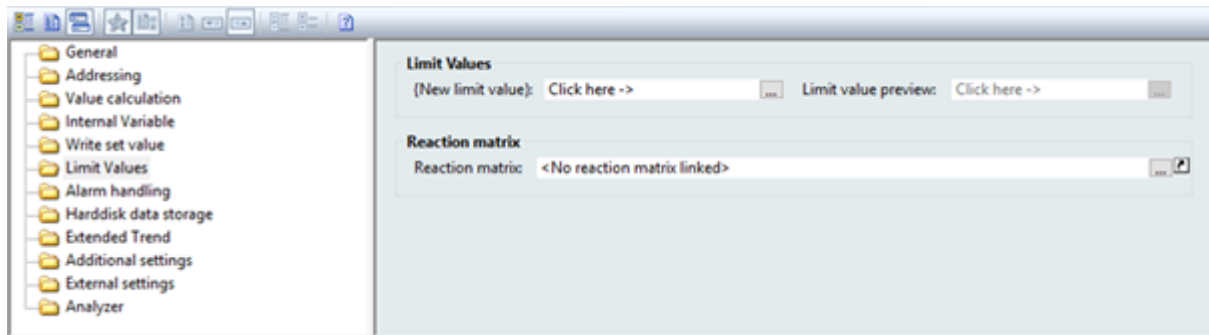
Any number of limit values can be defined per variable.

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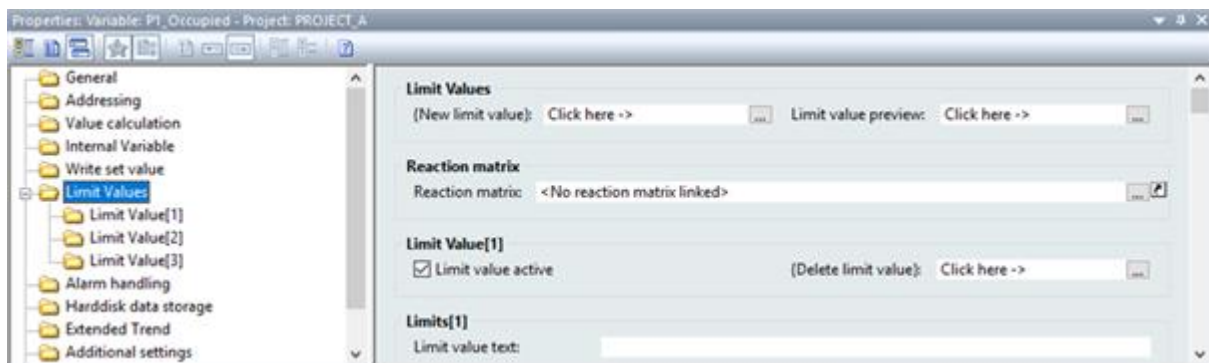
11.1.1 Delay

The **Delay time [s]** property can be used to configure a value for a time delay in the properties for the limit value. This value defined the time range in seconds in which the limit value violation must occur in order for the action defined for the limit value violation to be carried out.
Default: *0- no time delay*

Maximum value: *4294967295 seconds*

The delay time is calculated on the time stamp of the variable. It applies regardless from which way the limit was violated.

Example:

Engineering:

▸ **Limit value 1** has a value of *50*.

▸ **Limit value 2** has a value of *100*.

▸ The delay time is *3* s.

<u>**Procedure:**</u>

1. The value increases to *110*.
   After the violation has occurred for 3 seconds, the limit value violation for the value *100* is displayed.

2. The value goes down to *70*.
   As soon as this value has lasted for *3* seconds, the alarm for the limit value violation 100 is deactivated and a limit value violation for the value *50* is displayed.

3. If the value goes down to *40*, the alarm is deactivated as soon as there is no limit value violation any more.

TIME STAMP AND VALUE

The time of the first violation is entered as the time stamp for the limit value 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.
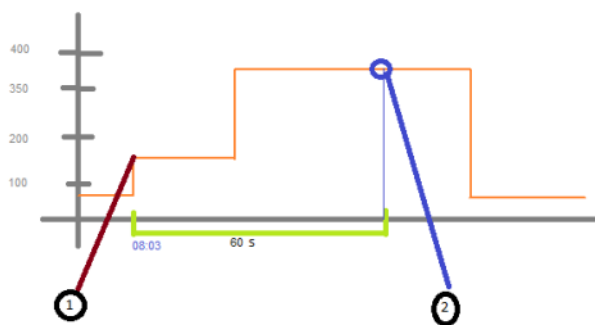
Key:

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

Digit	Description
	▸ Value displayed: Value after the delay time (2)

💡 **Information**

In combination with the **Flashing** property, this property has the following effect in the network:
Linked elements for limit values and alarms, each with or without delay, are shown on the server and on the clients as flashing.

11.1.2 Threshold

Thresholds prevent unwanted reactions in the event of quickly-changing (fluctuating) numerical values. They create a limit value hysteresis. The limit value is therefore a different one for the start of the limit value violation than the end of the limit value violation.

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

Thresholds can be set for maximum or minimum. For maximum, the threshold moves downwards, and upwards for minimum.
for example maximum value a maximum value of 900 and a threshold of 10 result in 890
The limit value will be violated when exceeding 900 and the critical area will be left when falling under 890.
Example minimum value: a minimum value of 900 and a threshold of 10 result in 910
The limit value will be violated when falling below 900 and the critical area will be left when exceeding 910.
As opposed to the hysteresis (on page 114), this property will only affect the alarm behavior and not the representation of the value.

11.1.3 Deduce limit values from datatypes

Limit values can already be defined in the datatype. The limit values 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 value also can be deactivated in a single variable. A limit value inherited from the datatype **cannot be deleted** in a single variable (deleting is only possible in the datatype). But further limit values can be added in the variable. These variable-specific limit values of course can be deleted in the variable.

More information on reaction matrices can be found in the chapter Datatypes (on page 15).

11.1.4 Multiselect

If several variables are selected at the same time, the properties of the limit values can also be amended for several variables at the same time. If one of the selected variables does not have the limit value 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. limit values), they are marked in yellow.

11.1.5 Deleting limit values

Click "delete limit value" in the properties.
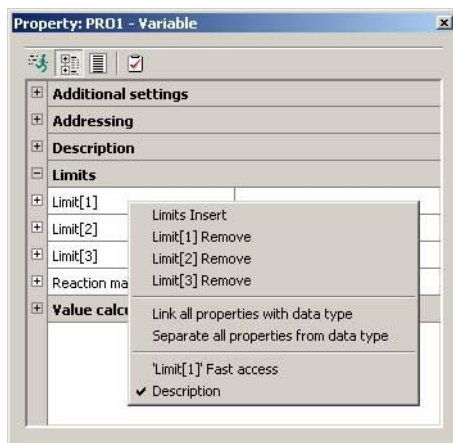
11.1.6 Overlapping limit values

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

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

11.1.7 Limit value preview

The limit values for a variable or a simple data type, as well as linked reaction matrices (on page 138), can be displayed in a preview dialog.
The display is called up for:

▶   Variables, simple data types and reaction matrices: for the limit values

▶   Screen elements: for the respective properties for colors, Glow, flashing or visibility, provided these are defined using variables
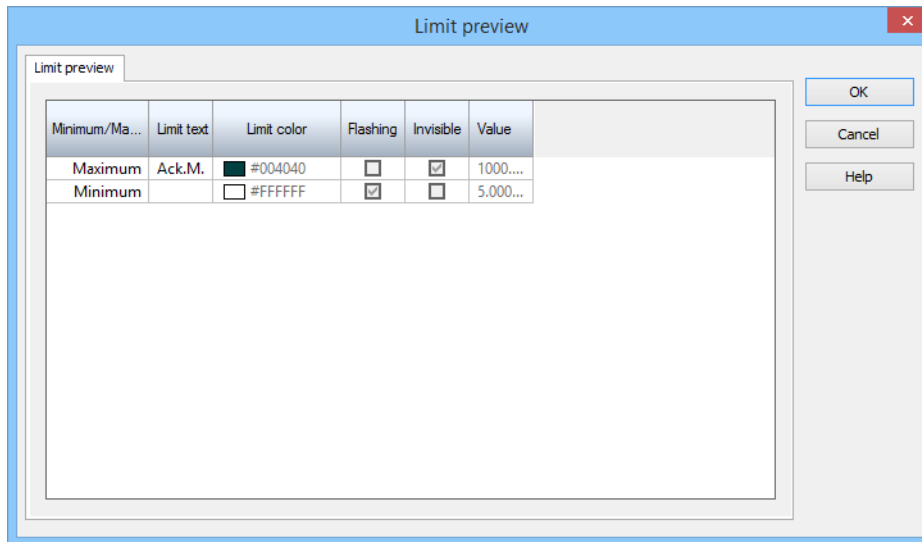
DISPLAY LIMIT VALUE PREVIEW

In order to display the limit value preview, open the dialog by means of the respective property:

Element	Property group	Property
Screen elements	Flashing	Show flash preview
Screen elements	Fill color(s) dynamic	Show color preview for Line color Show color preview for Fill color Show color preview for Gradient color
Screen elements	Glow (DirectX only)	Show color preview for Variable for color Display visibility preview for Variable for visibility Show flash preview for Variable for flashing
Screen elements	Visibility	Display visibility preview
Variables and simple data types	Limit Values	Limit value preview The corresponding dialog is called up depending on the type of limit value configuration, via limit values or reaction matrices: ▶ Limit value preview ▶ Preview of reaction matrix status

11.1.7.1 Limit value preview dialog

LIMIT VALUE PREVIEW DIALOG

The preview shows information on the configured limit values. The **Reaction matrix** property must not contain a link.

Parameters	Description
Limit value preview	Display of the defined limit value with the following properties: ▸ **Minimum/Maximum** ▸ **Limit value text** ▸ **Limit value color** ▸ **Flashing** ▸ **Invisible** ▸ **Limit Value** No configuration possible.
OK	Closes the dialog.
Cancel	Closes the dialog.
Help	Opens help

11.1.7.2Preview of reaction matrix status dialog

If a reaction matrix was selected for the limit values, then configured limit values are deactivated. The preview shows information about the linked reaction matrix.
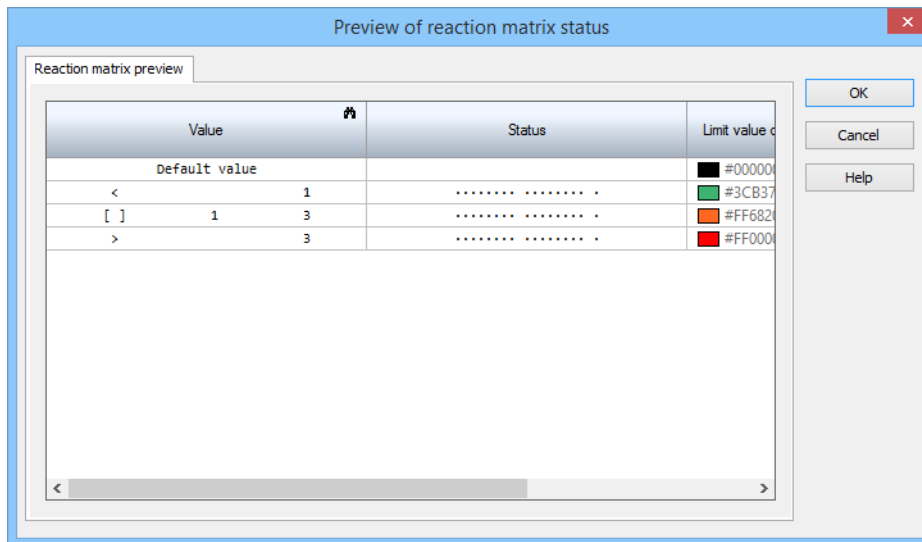
Parameters	Description
Reaction matrix preview	Display of the selected reaction matrix with the following information: ▸ Value ▸ Status ▸ Limit value color ▸ Invisible ▸ Flashing ▸ Limit value text No configuration possible.
OK	Closes the dialog.
Cancel	Closes the dialog.
Help	Opens help

11.2 Limit values 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 (GreaterThanEqualTo) and reaching or violating the defined minimum (LessThanEqualTo). This is called a limit value violation.

> ### ☀ Information
>
> A binary variable has two limit values, a Min limit and a Max limit. Care that:
> Enter the Min limit value as limit value 1, and the Max limit vamlue as limit value 2.

At the calculation of limit values 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:

PLC	Control system
0	0
127	50
255	100

If a limit value 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 value 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.

11.2.1 End of a limit value violation

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

For example: A limit value 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 value.

Unwanted reactions for quickly-changing (fluctuating) numerical values can be avoided by using thresholds (on page 119).

11.3 Dynamic limit value text

The dynamic limit value text makes it possible to include the current values of other variables, as well as additional variable information (on page 128) in the limit value text of a variable and in the limit value text of reaction matrices (on page 131). In doing so, a fixed text can be linked via an index with key words from a language table (see also Using key words in limit value texts (on page 130)).

The dynamic limit value text can be shown in:

- ▶ the Alarm Message List

- ▶ the Chronological Event List

- ▶ the Report Viewer

- ▶ The alarm status line

- ▶ The *dynamic text* element

> ☀ **Information**
>
> If the dynamic limit value is shown using the *dynamic text* element, the current variable value is always used.

SYNTAX

The following nomenclature is valid for the setting of the parameters:

Parameter	Description
$	Denotes a dynamic limit value text. This character must be on the first position in the limit value 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

Parameter	Description
	searched in the language table which is a combination of 'text' and variable value.
%Placeholder	For the display of additional information, see display of variable information (on page 128).

The using of combined keywords makes it possible to create dynamic limit value texts which can have more than 80 characters.

▤ Example 1

Limit value 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

LIMIT VALUES FOR DYNAMIC LIMIT VALUE TEXTS

The saving of dynamic limit value texts leads to the following limitations:

1. Static text is stored in the storage area of the limit value text:
   maximal 1024 characters

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

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

This limits can be evaded:

GETTING AROUND THE LIMITATION BY MEANS OF LONG DYNAMIC LIMIT VALUE TEXTS

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

Requirements: In the project properties in the **Alarm Message List** group, the **Long dynamic limit value texts AML** property, or in the **Chronological Event List group**, the **Long dynamic limit value texts CEL** property is active.
Instead of the commentary field the values are stored in an own file.

1032 bytes are available per limit value in principle. Of this, a certain number of bytes is deducted for each linked variable in the limit value text. The amount deducted depends on the data type. Therefore when several linked variables are used in the dynamic limit value text, there are also correspondingly less characters available.

Example 1: Dynamic limit value with a string variable:
Of the 1032 byte:

▸ 9 bytes are used for structural information

▸ 1023 bytes remain for the content of the string variable

Example 2: Dynamic limit value with two string variables:
Of the 1032 byte

▸ 18 (2 x 9) bytes are used for structural information

▸ 1014 bytes remain for the content of both string variables

Note:

▸ There is no limitation to the permitted number of variables per limit value text.
Caution: The sum of the bytes for the structure information and the bytes for the variable content must not exceed 1032 bytes.
Exceeding this can lead to loss of data.

▸ The number of bytes needed for the structure information depends on the data type.

▸ 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:

Parameter	Description
XXX	The value of a linked variable has changed.
---	values of linked variables are not available (can happen at program start).
>>>	Maximum number of characters for the dynamic limit value text has been exceeded.

11.3.1 Display of variable information

The following variable information can be shown with the help of the dynamic limit value text, using placeholders.

RESOURCES LABEL

Placeholder	Description
%r	Shows the resources label.
%r,1,2	Shows the resources label starting at position 1 for 2 characters.
%R	Shows the resources label.
%R,3,2	Shows the resources label minus the first 3 and the last 2 characters.

MEASURING UNIT

Placeholder	Description
%u	Shows the measuring unit.
%u,1,2	Shows the measuring unit starting at position 1 for 2 characters.
%U	Shows the measuring unit.
%U,3,2	Shows the measuring unit minus the first 3 and the last 2 characters

VARIABLE IDENTIFICATION

Placeholder	Description
%l	shows the variable identification.
%l,1,2	Shows the variable identification starting at position 1 for 2 characters.
%L	shows the variable identification.
%L,3,2	Shows the variable identification minus the first 3 and the last 2 characters.

VARIABLE NAME

Placeholder	Description
%n	shows the variable name.
%n,1,2	Shows the variable name starting at position 1 for 2 characters.
%N	shows the variable name.
%N,3,2	Shows the variable name minus the first 3 and the last 2 characters.

VARIABLE VALUE

Placeholder	Description
%v	Shows the current variable value.
%v,1,2	Shows the current variable value starting at position 1 for 2 characters.
%V	Shows the current variable value.
%V,3,2	Shows the current variable value minus the first 3 and the last 2 characters.

TIMESTAMP

Placeholder	Description
%t	Shows the time stamp.
%t,1,2	Shows the time stamp starting at position 1 for 2 characters.
	Shows the time stamp.
%T,3,2	Shows the time stamp minus the first 3 and the last 2 characters.

The formats for the time settings from the operating system are applied.

> 📄 **Example**
>
> ▶ Display of language-switchable text, the variable value and the measuring unit:
>
> *$@Text - ;%v; ;%u*
>
> ▶ Display of language-switchable text and the variable identification, limited to the first three characters:
>
> *$@Text - ;%l,1,3*

11.3.2 Dynamic key words in limit value texts

Alarm messages from Boolean variables do not provide the reason why an alarm was triggered when dealing with collective messages. So that the limit value text nevertheless expresses the cause of the alarm, the cause can be transferred using an index variable. The correct text is found and displayed using the dynamic limit value text (see Language switching chapter). For this, no fixed word can be used as a key word, but instead a dynamic one. This consists of **prefix+value of index variable**: *%@errornr+mIndex*

It is possible to use multi-dimensional key words (several indices).
For example: *fixed text; @MotorError%MyIndex1-%MyIndex2* becomes *@MotorError4-67* in the 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 value 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.

12 Reaction matrix

Limit value 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.

Note: You can search for non-used reaction matrixes with the project analysis. To do this, in the project tree, go to the Variables node and then the Reaction Matrix node. Open the context menu and click on the **Show unused reaction matrixes...** entry. Alternatively, you can also use the context menu in the reaction matrix editor. Used reaction matrixes can be searched for in this context menu using the **Reaction matrix usage...** entry. However, this is only possible once at least one reaction matrix from the list has been highlighted.

All entries found with the project analysis are displayed in their own separate project analysis window. As soon as a list entry has been highlighted, the linked jump target can be displayed using the **Jump to linked element** entry in the context menu.

ADVANTAGES WITH RESPECT TO LIMIT VALUES:

▸ 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 value violation. This is not possible in limit values.

▸ 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 limit values can be found in the chapter Limits (on page 116).

12.1 Creating a reaction matrix

Select **Reaction matrix** in the Project Manager and open the context menu with the right mouse button.

Select **New reaction matrix...**.

As an option, you can also use the **New reaction matrix...** entry in the tool bar (on page 136) to create a new reaction matrix.

In the dialog box that appears, enter the name and type of reaction matrix. This assignment can not be changed later. However the name can be changed later.
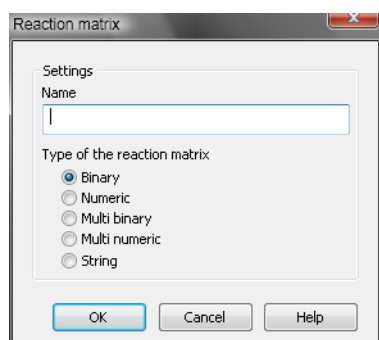
The individual types of reaction matrix are used for the following evaluations:

Parameter	Description
Binary	Simplified evaluation of discrete states (bit-orientated) for the compatibility to older zenon versions.
Numeric	Simplified evaluation of analog states (value-orientated) for the compatibility to older zenon versions.

Parameter	Description
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; evaluation of the status bits of the variable; special function (switch-off of variables when status bit is set)
Multi numeric	Extended evaluation of the value and the status bits of the variable; special function (switch-off of variables when status bit is set)
String	collation

12.2 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.
	Only available if a selection has been made.
Use of reaction matrices...	Opens the project analysis and shows the use of reaction matrices in a table.
	Table columns:
	▶ **Project**
	▶ **Module**
	▶ **Name**
	▶ **Visual name**
	▶ **Property**
	Only available if a selection has been made.
Display unused reaction matrices...	Opens the project analysis and shows the unused reaction matrices in a table.

Parameter	Description
	Table columns:
	▸ **Project**
	▸ **Module**
	▸ **Name**
	▸ **Visual name**
	▸ **Property**
Copy	Copies the selected entries to the clipboard.
	Only available if a selection has been made.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".
	Only available if there is content in the clipboard.
Delete	Deletes selected entries after a confirmation from list.
	Only available if a selection has been made.
Export selected as XML...	Exports all selected entries as an XML file.
	Only available if a selection has been made.
Import XML...	Imports entries from an XML file.
Remove all filters	Removes all filter settings.
	Note: Only active if the current view is filtered.
	Only available if at least one filter has been set.
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 cells that can be edited can be selected.
	Only available if a selection has been made.
Properties	Opens the **Properties** window.
	Only available if a selection has been made.
Help	Opens 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 state 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 to test 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:

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

12.2.1 Tool bar

Reaction matrices can be created and edited using the context menu (on page 133) or the tool bar.

Parameter	Description
New Reaction Matrix...	Creates a new reaction matrix.
Configuration...	Opens the dialog for configuring the reaction matrix. Only available if a selection has been made.
Use of reaction matrices...	Opens the project analysis and shows the use of reaction matrices in a table. Table columns: ▸ **Project** ▸ **Module** ▸ **Name** ▸ **Visual name** ▸ **Property** Only available if a selection has been made.
Display unused reaction matrices...	Opens the project analysis and shows the unused reaction matrices in a table. Table columns: ▸ **Project** ▸ **Module** ▸ **Name** ▸ **Visual name** ▸ **Property**

Parameter	Description
Jump back to starting element	Jumps back to the initial position in the zenon Editor. **Note:** Only available in the context menu if a jump to the current position has been made from another position with the **Linked elements** context menu entry.
Copy	Copies the selected entries to the clipboard. Only available if a selection has been made.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**". Only available if there is content in the clipboard.
Delete	Deletes selected entries after a confirmation from list. Only available if a selection has been made.
Export selected as XML...	Exports all selected entries as an XML file. Only available if a selection has been made.
Import XML...	Imports entries from an XML file.
Remove all filters	Removes all filter settings. Only available if at least one filter has been set.
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 cells that can be edited can be selected. Only available if a selection has been made.
Replace text in selected column...	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only cells that can be edited can be selected. Only available if a selection has been made.
Properties	Opens the **Properties** window.

Parameter	Description
	Only available if a selection has been made.
Help	Opens online help.

12.3 Link reaction matrices to variable or data type

Reaction matrices can be linked to variables or data types instead of limit values.

To link a reaction matrix:

1. Select the desired variable file or the desired data type.

2. Go to property group **Limit Values**\

3. In the **Reaction matrix** property, click on the **...** button.

4. The dialog to select and configure the reaction matrices is opened.

5. Select the desired reaction matrix.

6. Close the dialog by clicking on **OK**.

Note: To remove a reaction matrix that has already been linked, click on **None** in the dialog.
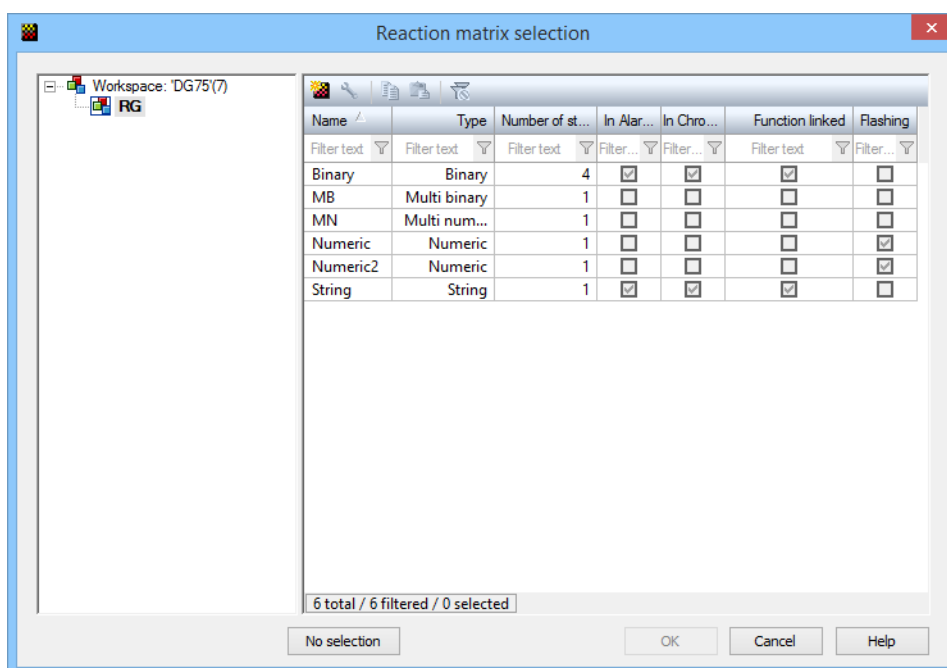

☀ Information

If you link a data type to a reaction matrix, all variables that have been newly created on the basis of this data type automatically use this reaction matrix.

REACTION MATRIX SELECTION DIALOG

The dialog allows the selection of a reaction matrix for a variable or a data type. The following are displayed for each existing reaction matrix:

▶ Name

▶ Type

▶ Number of configured states

▶ Apply in AML

▶ Apply in CEL

▶ Is a function linked

▶ Is flashing active or inactive

Reaction matrices can also be configured and newly created in this dialog. It is not possible to delete reaction matrices.

Parameter	Description
List of projects	Display of the active project. Only reaction matrices from this project can be linked.
List of reaction matrices	List of the reaction matrices that have already been configured in this project. Information on the name, type and configuration are displayed for each reaction matrix. Select by clicking on the reaction matrix and clicking on **OK**. The list can be shown after filtering. To do this, enter the filter term into the filter field. Placeholders ***** and **?** can be used. All reaction matrices can be configured, copied and newly created using the toolbar and the context menu. Reaction matrices can also be renamed. To rename a reaction matrix, double click slowly in the **Name** cell or highlight the cell and press the **F2 key**.
No selection	Clicking this removes a pre-existing linking and closes the dialog.

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

TOOLBAR AND CONTEXT MENU

Actions for reaction matrices in the **reaction matrix selection** dialog can be carried out using the toolbar or the context menu.

Symbol/Command	Description
New reaction matrix	Opens the dialog (on page 132) for creating a new reaction matrix.
Configuration	Opens the dialog for configuring the selected reaction matrix.
Copy	Copies the selected reaction matrix to the clipboard. **Note:** For copying, several reaction matrices can be selected at the same time.
Insert	Pastes copied reaction matrices into the list.
Remove all filters	Removes all configured filters.

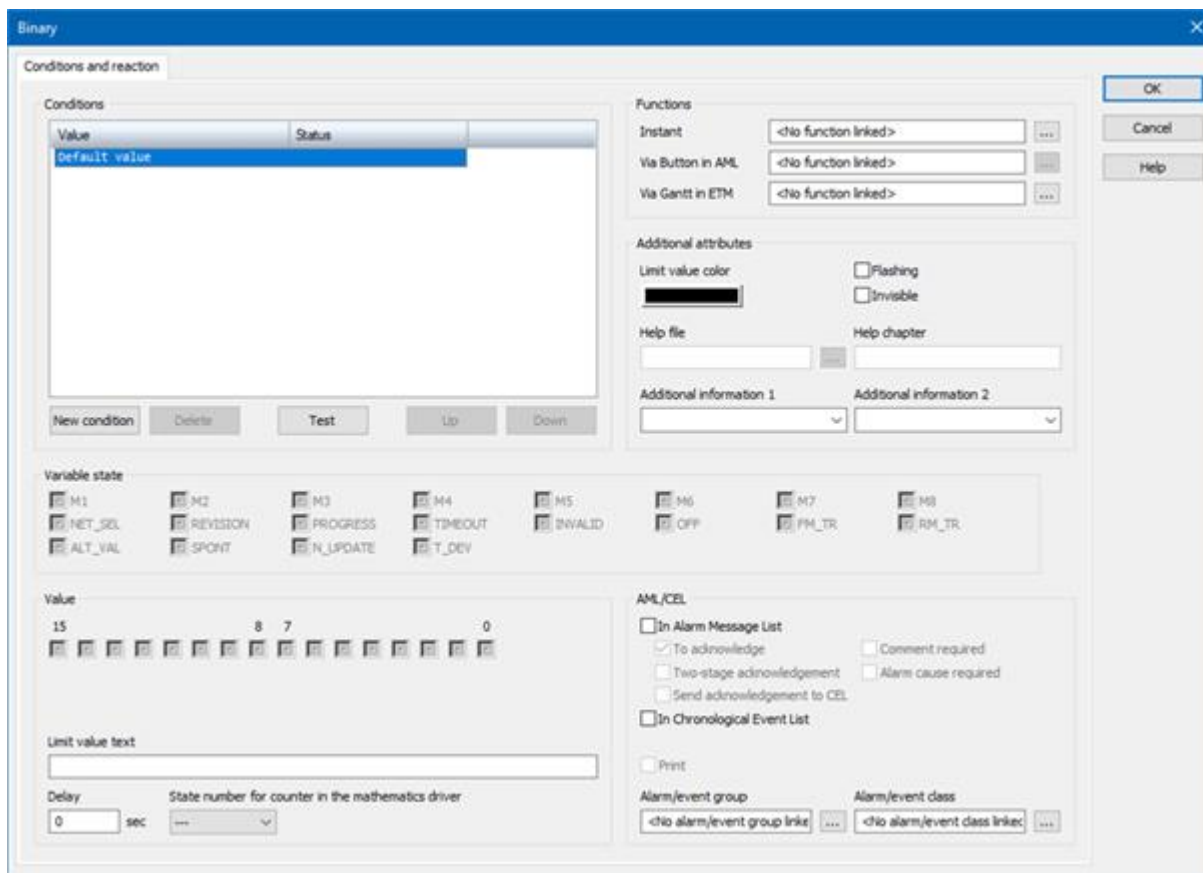## 12.4 Types of reaction matrices

There are five types of reaction matrices available:

> ▶ Binary (on page 141)
>
> ▶ Numeric (on page 148)
>
> ▶ String (on page 172)
>
> ▶ Multibinary (on page 157)
>
> ▶ Multi numeric (on page 162)

For multi-binary and multi-numeric types, AML/CEL text (on page 169), status routing (on page 169) and special functions (on page 171) can also be configured.

12.4.1 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.

STATES

Option	Description
Conditions list	List of the engineered states with value, state and cause of transmission.
New	Creates a new status based on the currently selected state.
Delete	Deletes all selected statues from the list.
Test	Opens the dialog for testing (on page 147) the condition.
Up	Moves selected states up.
Down	Moves selected states down.

VARIABLE STATE

Status bits selection.

Option	Description
State	Selection of the states via checkbox. Click in the checkbox in order to change the condition of the status. **Note:** For a list of all available status bits in zenon refer to manual Status processing chapter status bits. Not every status bit can be evaluated by means of reaction matrices.

VALUE

Option	Description
Value	Current value of variable.
Limit value text	Text which is displayed when at a limit value violation. Output in: ▸ dynamic text element ▸ CEL ▸ AML ▸ ...
Delay	Time period which the limit value violation must last in order for the limit value to be active.
State number for counter in the mathematics driver	Assignment of four possible status numbers for a counter in the mathematics driver. For this the variable must have a reaction matrix.

The definition of the combination of state 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.

FUNCTION

Option	Description
Immediately	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog.

Option	Description
	The linked function is executed immediately if there is a limit value violation of the reaction matrix variable.
With button in AML	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog. Only available if you activated option **In Alarm Message List** in **AML/CEL**. The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the corresponding entry has been selected in the **Alarm Message List (AML)** and the **Execute Function** button is clicked on.
With button in Gantt	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog. The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the Gantt curve is clicked on with the mouse pointer in the area of the limit value violation in the Runtime in the **Extended Trend (ETM)**.

ADDITIONAL ATTRIBUTES

Option	Description
Limit value color	Color when a limit value has been violated.
Flashing	▸ *Active*: Flashes when a limit value has been violated.
Invisible	▸ *Active*: Will be switch to invisible when a limit value 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**. The linked help file is opened in the Runtime if: ▸ A help chapter has been entered ▸ The corresponding alarm is selected in the Alarm

Option	Description
	Message list and
	‣ The **Open 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. Must contain an entry in order for help to be opened in the Runtime. **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

Option	Description
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.
Two-stage acknowledgment	Only available if **In Alarm Message List** is activated. ‣ *Active*: **Two-stage acknowledgement** is required.
Alarm cause required	Only available if **In Alarm Message List** is activated. ‣ *Active*: **Alarm cause** is required
In Chronological Event List	‣ *Active*: Will be entered in the CEL. **Note**: 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 value violation has been rectified and then is violated again, or the state becomes inactive and then active

Option	Description
	again, is a CEL entry generated.
Print	Only available if **In Alarm Message List** or in **In Chronological Event List** is activated. ▸ *Active*: Will be printed via the set standard printer.
Alarm/Event Group	Allocation to an alarm/event group. Select from drop-down list.
Alarm/Event Class	Allocation to an alarm/event class. Allocation to an alarm/event group. Select from drop-down list.

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

BIT ASSIGNMENT

Condition:	Bit assignment
Non-scaled value bit set "0"	$2^0 = 1$
Non-scaled value bit set "1"	$2^1 = 2$
Non-scaled value bit set "2"	$2^2 = 4$ ....

Two linkages are possible. The meaning of the fields is

Condition:	Description
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 value 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.
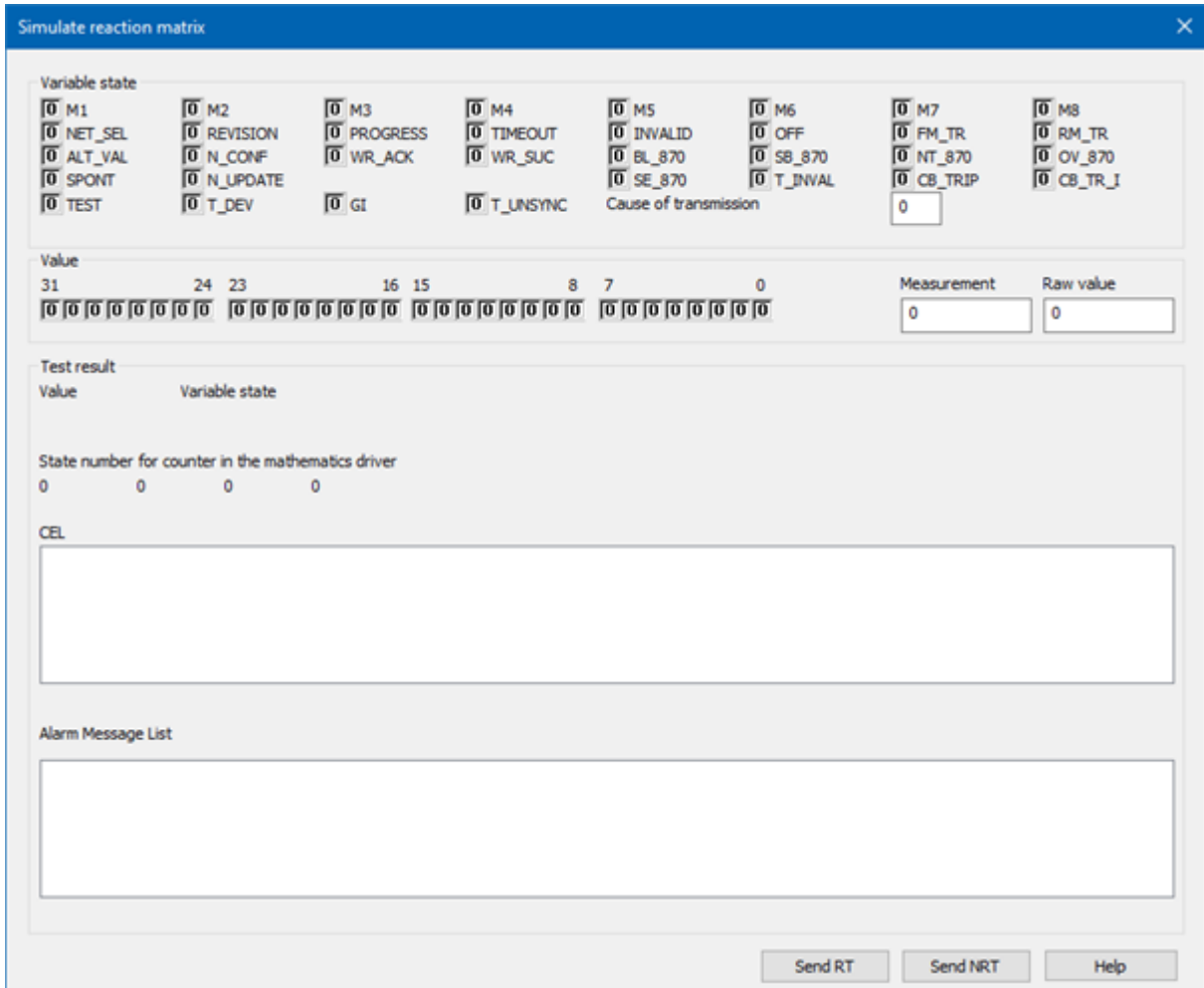
ANALYSIS

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 message with disturbed monitoring)
check INVALID-bit set	……………, …………1…
check bit0 set	……………0, ……………
check bit1 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, flashing, 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**.
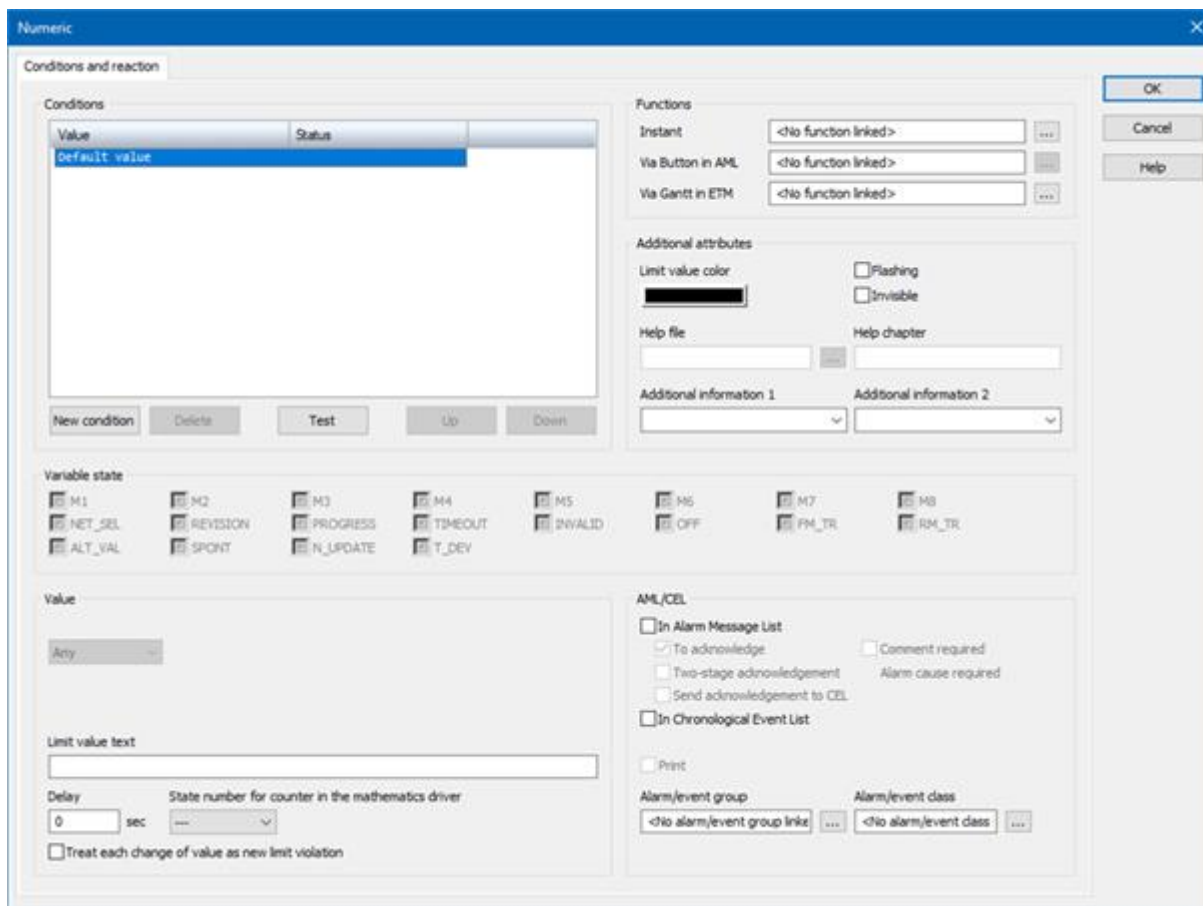
12.4.1.1 Test

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.

Option	Description
Variable state	To simulate status changes.
Value	Simulation of value changes.
Test result	Output of changes from status and value in CEL and AML.
Send RT	Send changes in real time.
Send NRT	Send changes.

12.4.2 Numeric

A **numeric reaction matrix** is used for the evaluation of any kind of limit value 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.

STATES

Option	Description
Conditions list	List of the engineered states with value, state and cause of transmission.
New	Creates a new status based on the currently selected state.
Delete	Deletes all selected statues from the list.
Test	Opens the dialog for testing (on page 147) the condition.
Up	Moves selected states up.

Option	Description
Down	Moves selected states down.

VARIABLE STATE

Status bits selection.

Option	Description
State	Selection of the states via checkbox. Click in the checkbox in order to change the condition of the status. **Note:** For a list of all available status bits in zenon refer to manual Status processing chapter status bits. Not every status bit can be evaluated by means of reaction matrices.

VALUE

Option	Description
Value	Definition of the declaration of value. The value is entered in the input field next to the stipulated value given. Select from drop-down list: ▸ *any*: any change of value violates the limit value ▸ *greater*: Input of a limit value and a threshold value. The value to be triggered must be greater than the value selected. The following is applicable for *String* reaction matrices: The further back in the alphabet, the greater. Small letters are *greater* than capital letters. ▸ *smaller*: Input of a limit value and a threshold value. The value to be triggered must be less than the value selected. The following is applicable for *String* reaction matrices: The further forward in the alphabet, the smaller. Capital letters are *less* than small letters. ▸ *equal*: Entry of a limit value and a threshold value.

Option	Description
	The value to be triggered must be exactly the same value.
	▸ *Range*: Enter a range (from/to) and a threshold value.
	Note: Only values of numeric data types can be displayed in the **Value** column for AML and CEL. The column remains empty for entries with a **String** data type. However, **String** values can be displayed using a *dynamic text* (on page 178) element.
Match case	Evaluation takes capitalization into account in text.
	Note: Text is configured in the input field next to the stated value.
	▸ *Active:* Evaluation takes capitalization into account. The text entered must correspond to the prescribed value exactly in order for the status to be valid.
	▸ Inactive: Evaluation does not take capitalization into account.
	Note: Only available for *string* reaction matrices.
Take wildcards into account	Evaluation takes placeholders into account.
	Note: Wildcards are configured in the input field next to the stated value.
	▸ *Active:* Evaluation takes wildcards into account.
	▸ *Inactive*: Evaluation does not take wildcards into account. The values must correspond to the prescribed value exactly in order for a status to be valid.
	Wildcards:
	▸ *****: Replaces desired characters in the desired quantity. Can be used as a search term at any desired place. **red*** finds all texts that start with **red**.
	▸ **?**: Replaces precisely one character. **r?d** finds **red**, **rad**, ...

Option	Description
	Note: Only available for *string* reaction matrices.
Limit value text	Text which is displayed when at a limit value violation. Output in: ▸ dynamic text element ▸ CEL ▸ AML ▸ ...
Delay	Time period which the limit value violation must last in order for the limit value to be active.
State number for counter in the mathematics driver	Assignment of four possible status numbers for a counter in the mathematics driver. For this the variable must have a reaction matrix.
Treat each change of value as new limit violation	▸ *Active:* Each value change in the range of an engineered condition is displayed as separate violation (CEL, AML, ...).

FUNCTION

Option	Description
Immediately	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog. The linked function is executed immediately if there is a limit value violation of the reaction matrix variable.
With button in AML	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog. Only available if you activated option **In Alarm Message List** in **AML/CEL**. The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the corresponding entry has been selected in the **Alarm Message List (AML)** and the **Execute Function** button is clicked on.

Option	Description
With button in Gantt	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog.
	The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the Gantt curve is clicked on with the mouse pointer in the area of the limit value violation in the Runtime in the **Extended Trend (ETM)**.

ADDITIONAL ATTRIBUTES

Option	Description
Limit value color	Color when a limit value has been violated.
Flashing	▸ *Active*: Flashes when a limit value has been violated.
Invisible	▸ *Active*: Will be switch to invisible when a limit value 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**.
	The linked help file is opened in the Runtime if:
	▸ A help chapter has been entered
	▸ The corresponding alarm is selected in the Alarm Message list and
	▸ The **Open 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. Must contain an entry in order for help to be opened in the Runtime.
	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

Option	Description
	assessed in a VBA macro.

AML/CEL

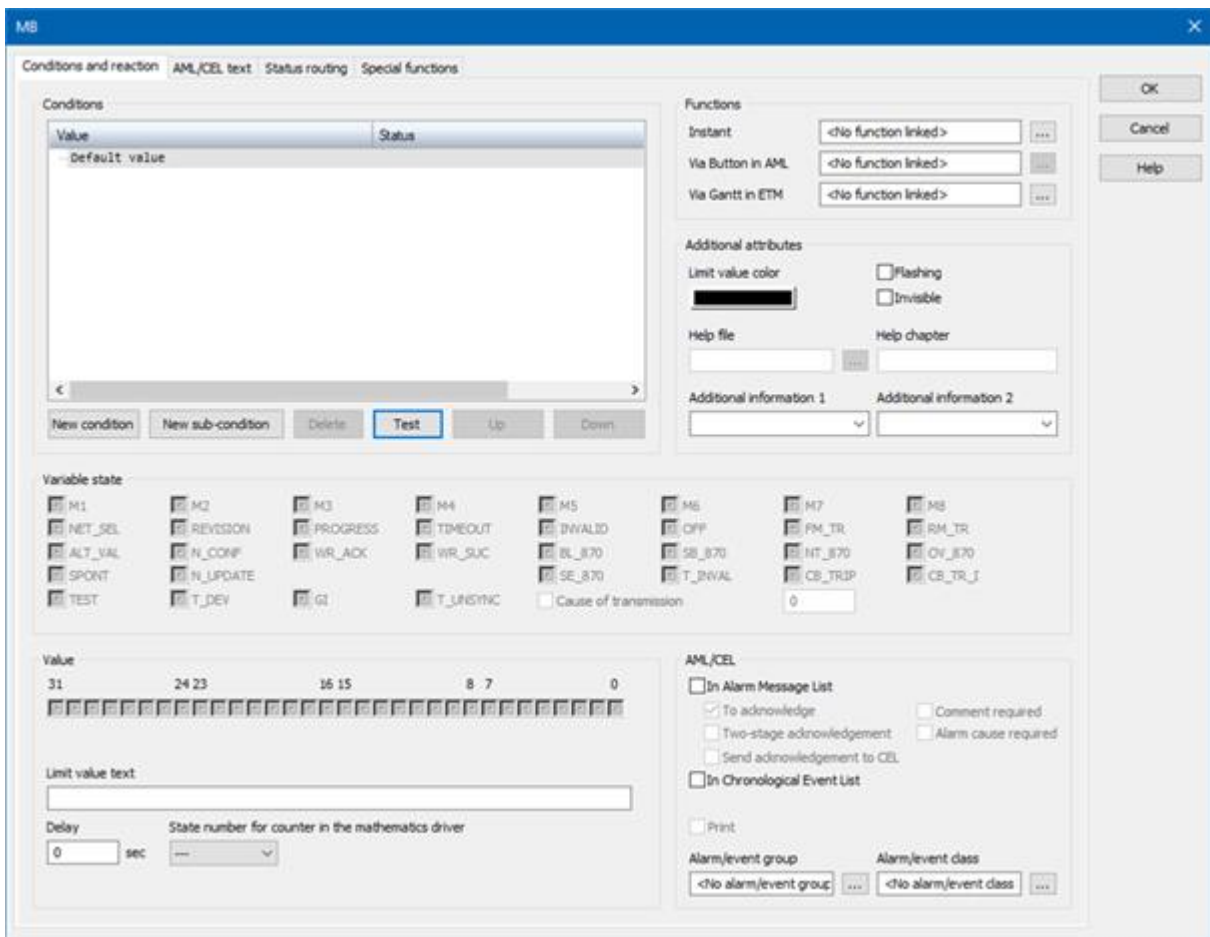| Option	Description
In Alarm Message List	▸ *Active*: Will be entered in the AML.
To acknowledge	Only available if **In Alarm Message List** is activated. ▸ *Active*: Must be acknowledged.
Comment required	To be able to acknowledge the alarm, a comment must be entered beforehand. The user must be authorized to carry out the necessary function.
Two-stage acknowledgment	Only available if **In Alarm Message List** is activated. ▸ *Active*: **Two-stage acknowledgement** is required.
Alarm cause required	Only available if **In Alarm Message List** is activated. ▸ *Active*: **Alarm cause** is required
In Chronological Event List	▸ *Active*: Will be entered in the CEL. **Note**: 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 value violation has been rectified and then is violated again, or the state becomes inactive and then active again, is a CEL entry generated.
Print	Only available if **In Alarm Message List** or in **In Chronological Event List** is activated. ▸ *Active*: Will be printed via the set standard printer.
Alarm/Event Group	Allocation to an alarm/event group. Select from drop-down list.
Alarm/Event Class	Allocation to an alarm/event class. Allocation to an alarm/event group. Select from drop-down list.

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

> ### 💡 Information
>
> 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**.

12.4.3 Multi-reaction matrices in general

Tabs	Description
States and reactions	Configuration for the states and reactions for the Multi-binary (on page 157) reaction matrix and the Multi-numeric (on page 162) reaction matrix.
AML/CEL text (on page 169)	Configuration for the output of the value next to the limit text in the AML and CEL.
Status routing (on page 169)	Configuration of the status routing.
Special functions (on page 171)	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:

Option	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

Option	Description
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 value 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 matrices enable the evaluation of the transmission cause based on IEC 60870-5-101/104 standard.
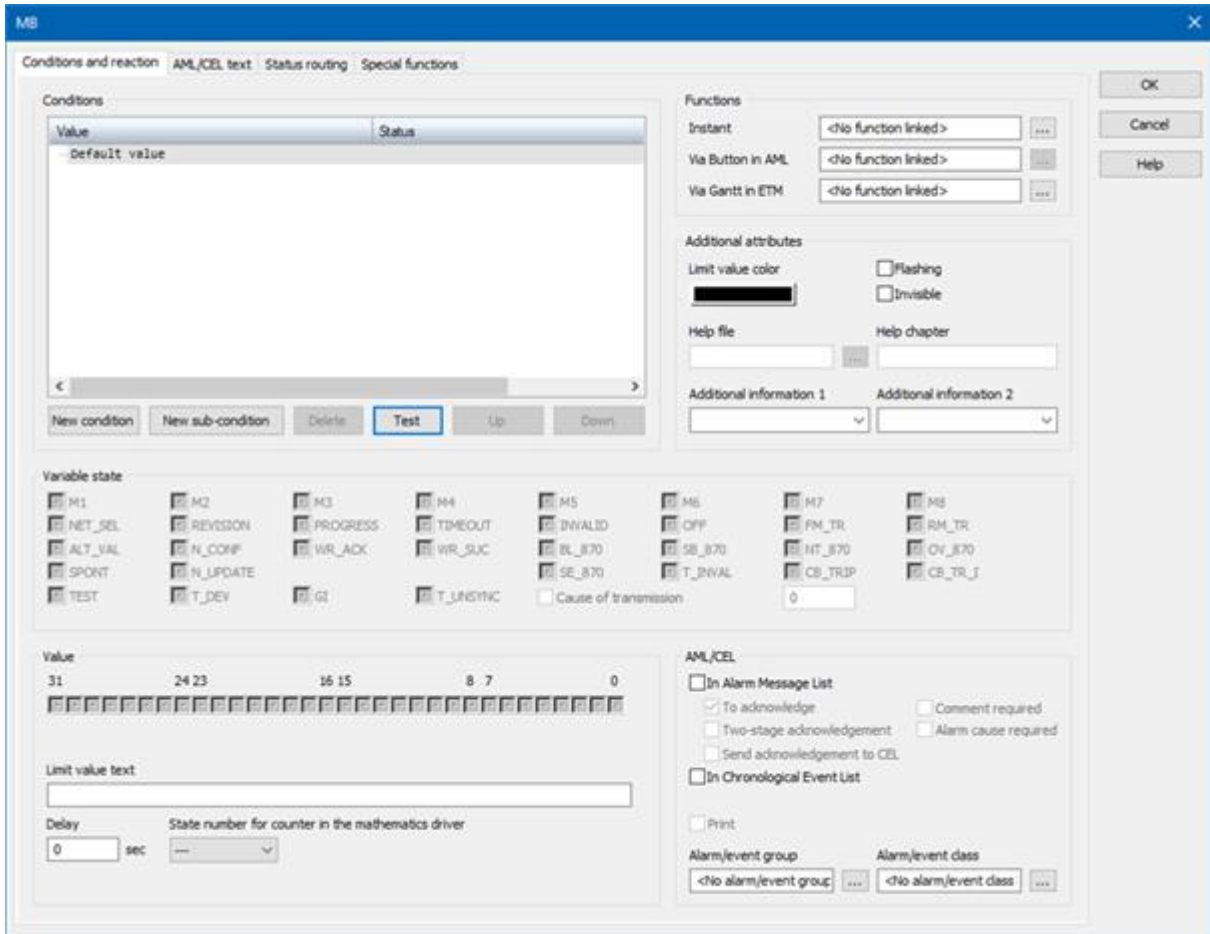
The transmission cause consists of:

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.

Option	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.

12.4.3.1 Multi-binary states and reactions

A **multi binary 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.

STATES

List of the engineered variable conditions with value, state and cause of transmission.

Option	Description
New condition	Creates a new status based on the currently selected state.
New sub-condition	Only available for mulit binary. Creates a new sub-status for the selected status.
Delete	Deletes all selected statues from the list.
Test	Opens the dialog (on page 147)for testing the status.

Option	Description
Up	Moves selected states up.
Down	Moves selected states down.

VARIABLE STATE

Status bits selection.

Option	Description
State	Selection of the states via checkbox. Click in the checkbox in order to change the condition of the status. **Note:** For a list of all available status bits in zenon refer to manual Status processing chapter status bits. Not every status bit can be evaluated by means of reaction matrices.
Cause of transmission	Additional status for cause of transmission (COT) in accordance with the IEC60870 protocol. For details, see the **Cause of transmission** chapter in the **Status processing** manual.

VALUE

Option	Description
Value	Current value of variable.
Limit value text	Text which is displayed when at a limit value violation. Output in: ▸ dynamic text element ▸ CEL ▸ AML ▸ ...
Delay	Time period which the limit value violation must last in order for the limit value to be active.
State number for counter in the mathematics driver	Assignment of four possible status numbers for a counter in the mathematics driver. For this the variable must have a reaction matrix.

The definition of the combination of state 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.

CHECKBOX STATUS AND VALUE

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

Possible conditions:

▶ *not considered* (dot)

▶ *on* (1)

▶ *off* (0)

▶ *coming* (arrow upwards)

▶ *going* (arrow downwards)

▶ *alternating* (dual arrow)

💡 Information

The statuses *coming, going* and *alternating* act differently to the standard statuses:

▶ The statuses only trigger for a moment.

▶ Further **Rema** statuses are checked after triggering.

▶ The initial value does not trigger these statuses.

FUNCTION

Option	Description
Immediately	Click on **…** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog. The linked function is executed immediately if there is a limit value violation of the reaction matrix variable.
With button in AML	Click on **…** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog. Only available if you activated option **In Alarm Message List** in **AML/CEL**. The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the corresponding entry has been selected in the **Alarm Message List (AML)** and the **Execute Function** button is

Option	Description
	clicked on.
With button in Gantt	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog.
	The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the Gantt curve is clicked on with the mouse pointer in the area of the limit value violation in the Runtime in the **Extended Trend (ETM)**.

ADDITIONAL ATTRIBUTES

Option	Description
Limit value color	Color when a limit value has been violated.
Flashing	▸ *Active*: Flashes when a limit value has been violated.
Invisible	▸ *Active*: Will be switch to invisible when a limit value 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**.
	The linked help file is opened in the Runtime if:
	▸ A help chapter has been entered
	▸ The corresponding alarm is selected in the Alarm Message list and
	▸ The **Open 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. Must contain an entry in order for help to be opened in the Runtime.
	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.

Option	Description
Additional information 2	In the Runtime the additional information entered can be assessed in a VBA macro.

AML/CEL

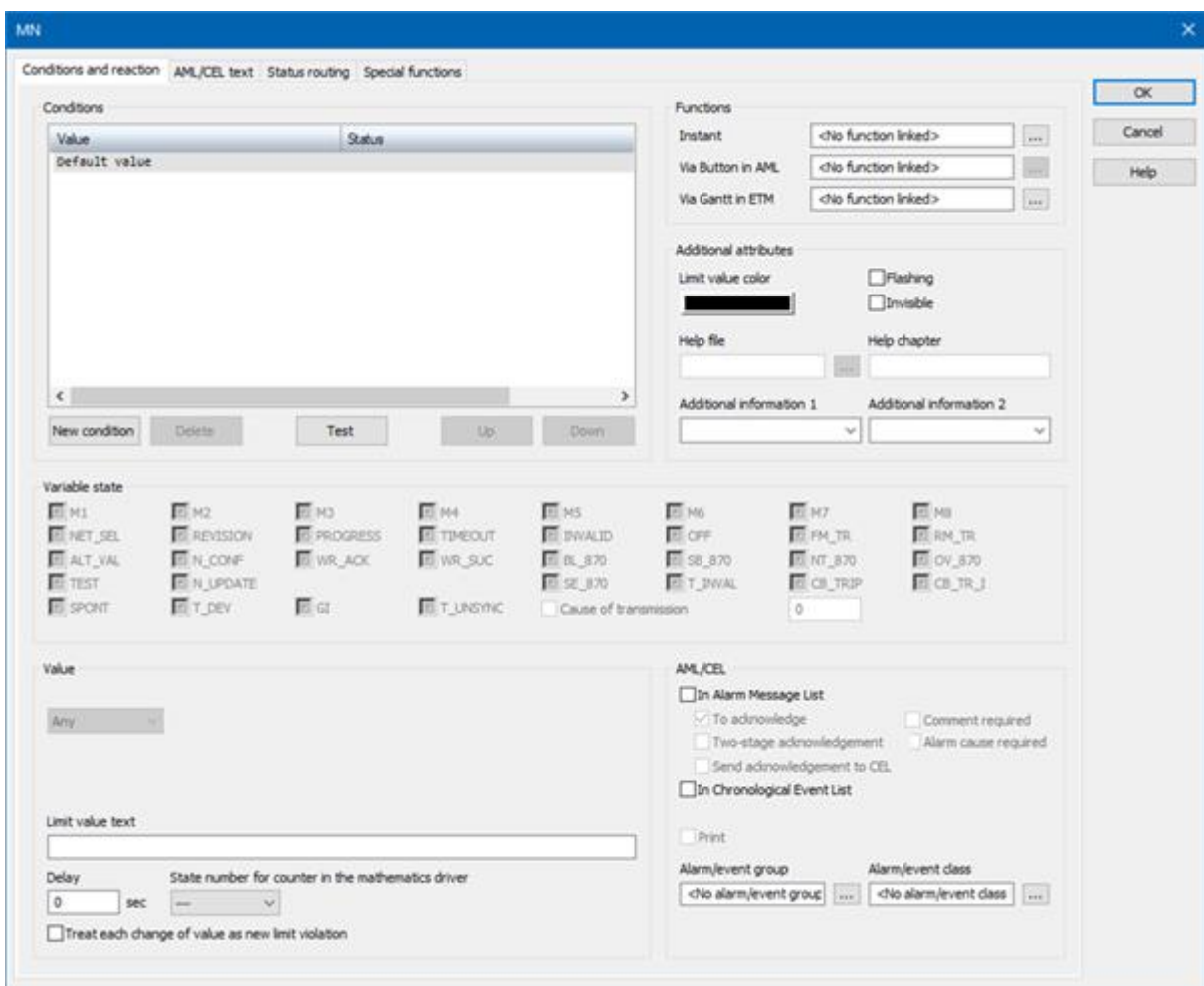| Option	Description
In Alarm Message List	▶ *Active*: Will be entered in the AML.
To acknowledge	Only available if **In Alarm Message List** is activated. ▶ *Active*: Must be acknowledged.
Comment required	To be able to acknowledge the alarm, a comment must be entered beforehand. The user must be authorized to carry out the necessary function.
Two-stage acknowledgment	Only available if **In Alarm Message List** is activated. ▶ *Active*: **Two-stage acknowledgement** is required.
Alarm cause required	Only available if **In Alarm Message List** is activated. ▶ *Active*: **Alarm cause** is required
In Chronological Event List	▶ *Active*: Will be entered in the CEL. **Note**: 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 value violation has been rectified and then is violated again, or the state becomes inactive and then active again, is a CEL entry generated.
Print	Only available if **In Alarm Message List** or in **In Chronological Event List** is activated. ▶ *Active*: Will be printed via the set standard printer.
Alarm/Event Group	Allocation to an alarm/event group. Select from drop-down list.
Alarm/Event Class	Allocation to an alarm/event class. Allocation to an alarm/event group. Select from drop-down list.

> 💡 **Information**
>
> 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**.

12.4.3.2 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.

STATES

List of the engineered variable conditions with value, state and cause of transmission.

Option	Description
New condition	Creates a new status based on the currently selected state.
New sub-condition	Only available for mulit binary. Creates a new sub-status for the selected status.
Delete	Deletes all selected statues from the list.
Test	Opens the dialog (on page 147)for testing the status.
Up	Moves selected states up.
Down	Moves selected states down.

VARIABLE STATE

Status bits selection.

Option	Description
State	Selection of the states via checkbox. Click in the checkbox in order to change the condition of the status. **Note:** For a list of all available status bits in zenon refer to manual Status processing chapter status bits. Not every status bit can be evaluated by means of reaction matrices.
Cause of transmission	Additional status for cause of transmission (COT) in accordance with the IEC60870 protocol. For details, see the **Cause of transmission** chapter in the **Status processing** manual.

VALUE

Option	Description
Value	Definition of the declaration of value. The value is entered in the input field next to the stipulated value given. Select from drop-down list: ▶ *any*: any change of value violates the limit value ▶ *greater*: Input of a limit value and a threshold value. The value to be triggered must be greater than the value selected.

Option	Description
	The following is applicable for *String* reaction matrices: The further back in the alphabet, the greater. Small letters are *greater* than capital letters. ▸ *smaller*: Input of a limit value and a threshold value. The value to be triggered must be less than the value selected. The following is applicable for *String* reaction matrices: The further forward in the alphabet, the smaller. Capital letters are *less* than small letters. ▸ *equal*: Entry of a limit value and a threshold value. The value to be triggered must be exactly the same value. ▸ *Range*: Enter a range (from/to) and a threshold value. **Note:** Only values of numeric data types can be displayed in the **Value** column for AML and CEL. The column remains empty for entries with a **String** data type. However, **String** values can be displayed using a *dynamic text* (on page 178) element.
Match case	Evaluation takes capitalization into account in text. **Note:** Text is configured in the input field next to the stated value. ▸ *Active:* Evaluation takes capitalization into account. The text entered must correspond to the prescribed value exactly in order for the status to be valid. ▸ Inactive: Evaluation does not take capitalization into account. **Note:** Only available for *string* reaction matrices.
Take wildcards into account	Evaluation takes placeholders into account. **Note:** Wildcards are configured in the input field next to the stated value. ▸ *Active:* Evaluation takes wildcards into account. ▸ *Inactive*: Evaluation does not take wildcards into

Option	Description
	account. The values must correspond to the prescribed value exactly in order for a status to be valid.
	Wildcards:
	▸ *****: Replaces desired characters in the desired quantity. Can be used as a search term at any desired place. **red*** finds all texts that start with **red**.
	▸ **?**: Replaces precisely one character. **r?d** finds **red**, **rad**, ...
	Note: Only available for *string* reaction matrices.
Limit value text	Text which is displayed when at a limit value violation.
	Output in:
	▸ dynamic text element
	▸ CEL
	▸ AML
	▸ ...
Delay	Time period which the limit value violation must last in order for the limit value to be active.
State number for counter in the mathematics driver	Assignment of four possible status numbers for a counter in the mathematics driver. For this the variable must have a reaction matrix.
Treat each change of value as new limit violation	▸ *Active:* Each value change in the range of an engineered condition is displayed as separate violation (CEL, AML, ...).

FUNCTION

Option	Description
Immediately	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog.
	The linked function is executed immediately if there is a limit value violation of the reaction matrix variable.

Option	Description
With button in AML	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog. Only available if you activated option **In Alarm Message List** in **AML/CEL**. The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the corresponding entry has been selected in the **Alarm Message List (AML)** and the **Execute Function** button is clicked on.
With button in Gantt	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog. The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the Gantt curve is clicked on with the mouse pointer in the area of the limit value violation in the Runtime in the **Extended Trend (ETM)**.

ADDITIONAL ATTRIBUTES

Option	Description
Limit value color	Color when a limit value has been violated.
Flashing	‣ *Active*: Flashes when a limit value has been violated.
Invisible	‣ *Active*: Will be switch to invisible when a limit value 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**. The linked help file is opened in the Runtime if: ‣ A help chapter has been entered ‣ The corresponding alarm is selected in the Alarm Message list and ‣ The **Open help** button is clicked

Option	Description
	Note: The property can only be configured if the **In Alarm Message List** property is active.
Help chapter	Indication of the help chapter. Must contain an entry in order for help to be opened in the Runtime. **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

Option	Description
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.
Two-stage acknowledgment	Only available if **In Alarm Message List** is activated. ‣ *Active*: **Two-stage acknowledgement** is required.
Alarm cause required	Only available if **In Alarm Message List** is activated. ‣ *Active*: **Alarm cause** is required
In Chronological Event List	‣ *Active*: Will be entered in the CEL. **Note**: 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 value violation has been rectified and then is violated again, or the state becomes inactive and then active again, is a CEL entry generated.
Print	Only available if **In Alarm Message List** or in **In**

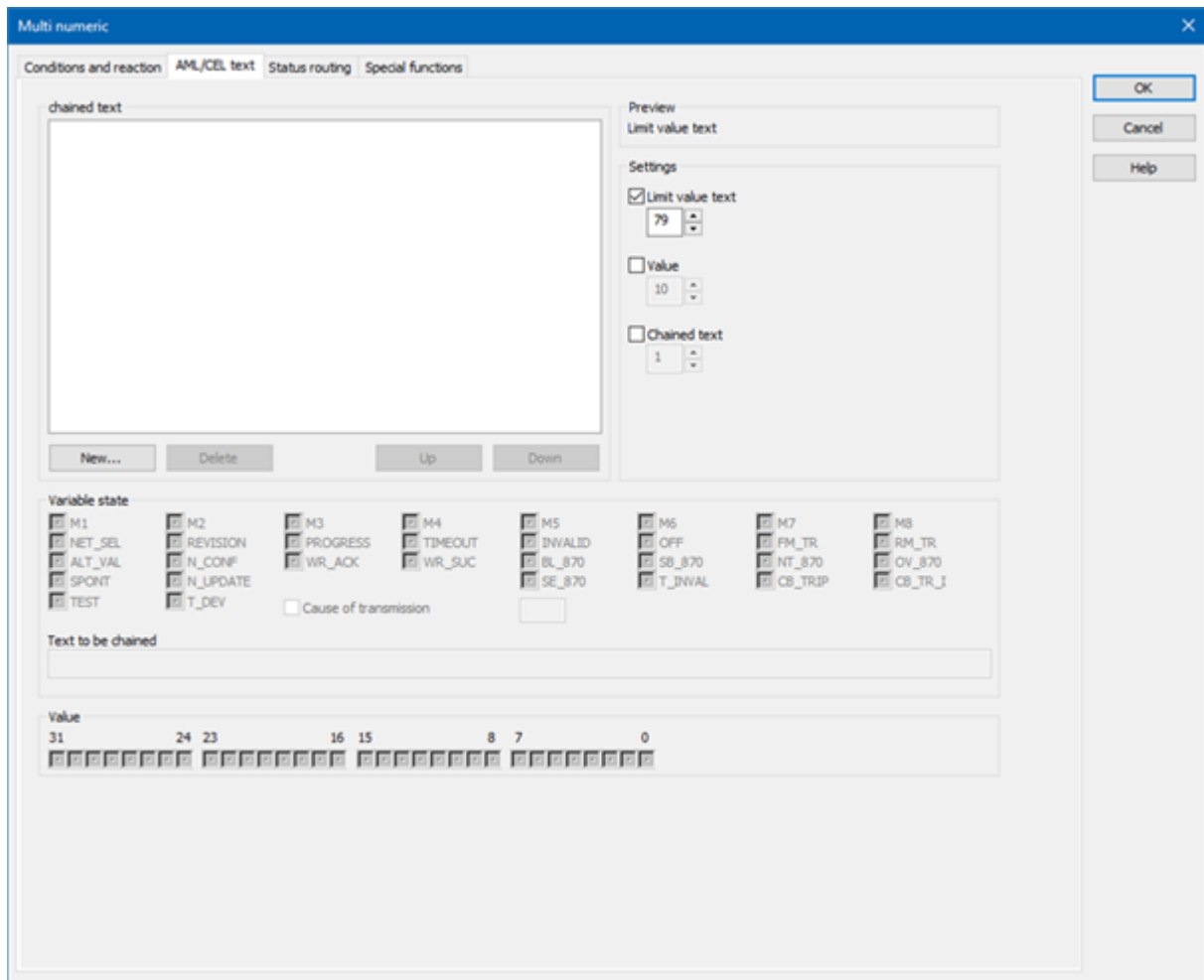| Option	Description
	Chronological Event List is activated. ▸ *Active*: Will be printed via the set standard printer.
Alarm/Event Group	Allocation to an alarm/event group. Select from drop-down list.
Alarm/Event Class	Allocation to an alarm/event class. Allocation to an alarm/event group. Select from drop-down list.

💡 **Information**

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**.
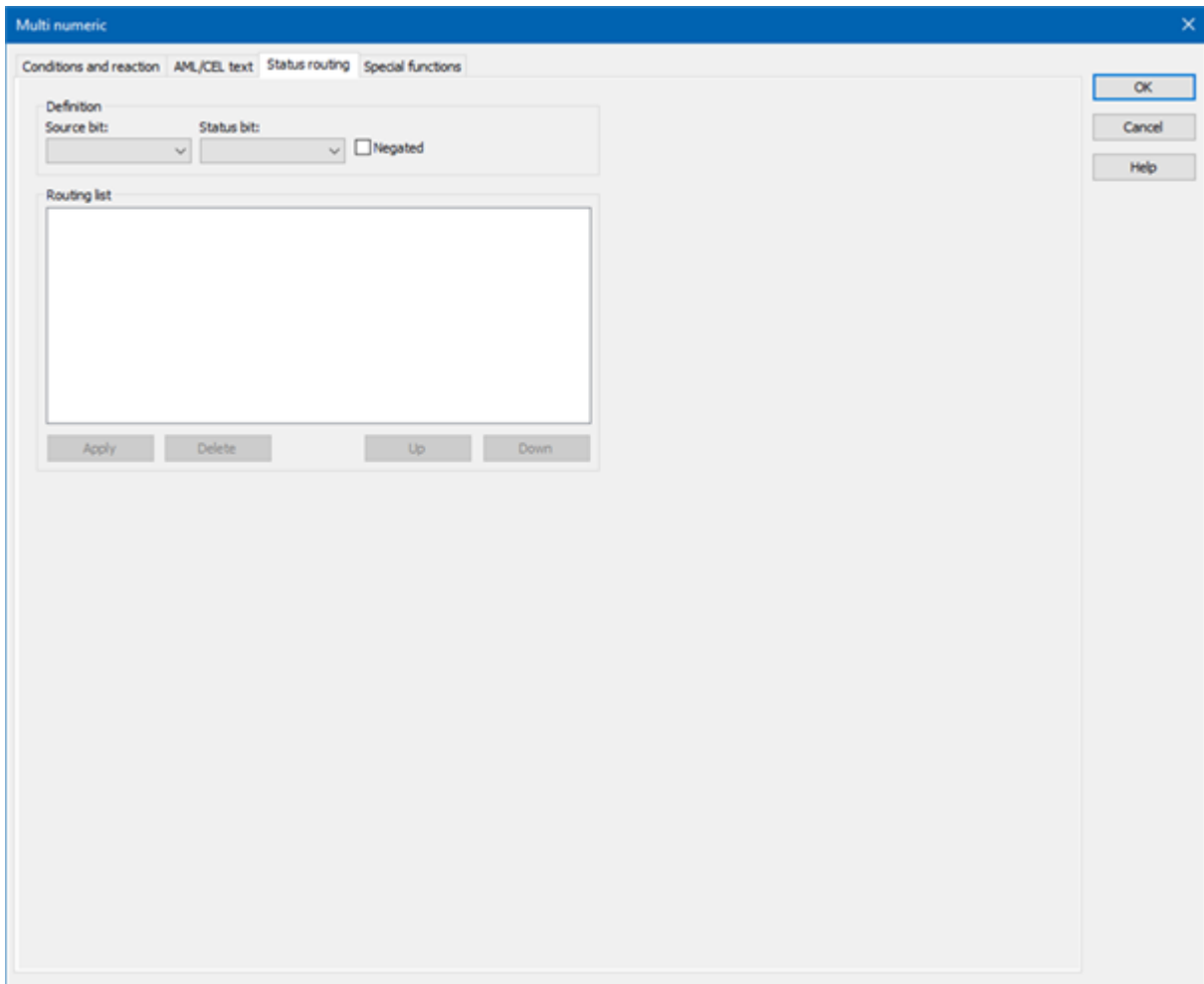
12.4.3.3 Configuration AML/CEL text

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

12.4.3.4 Configuration status routing

In this tab the status routing is configured. Routes bits (also negated ones) to other bits.
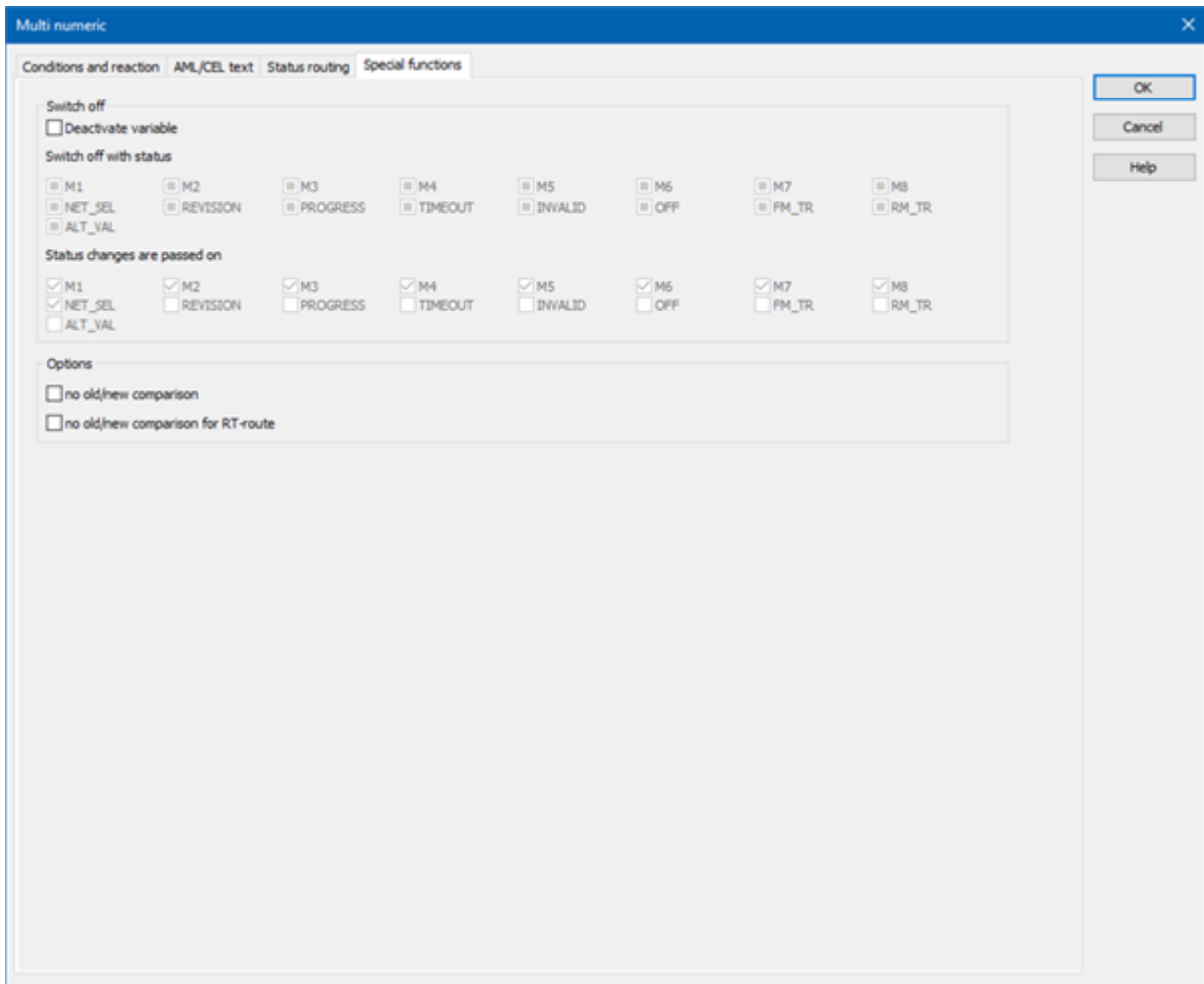
Possible sources: Value bit 0 ... 31l

Option	Description
Source bit	Select source bit from list box
NEG	Negate source bit during the routing
Status bit	Select target bit from list box
Apply	accept selected routing
Delete	delete selected routing; Attention: There is no further query
Up	move selected routing in the sequence upwards
Down	move selected routing in the sequence downwards

12.4.3.5 Configuration special functions

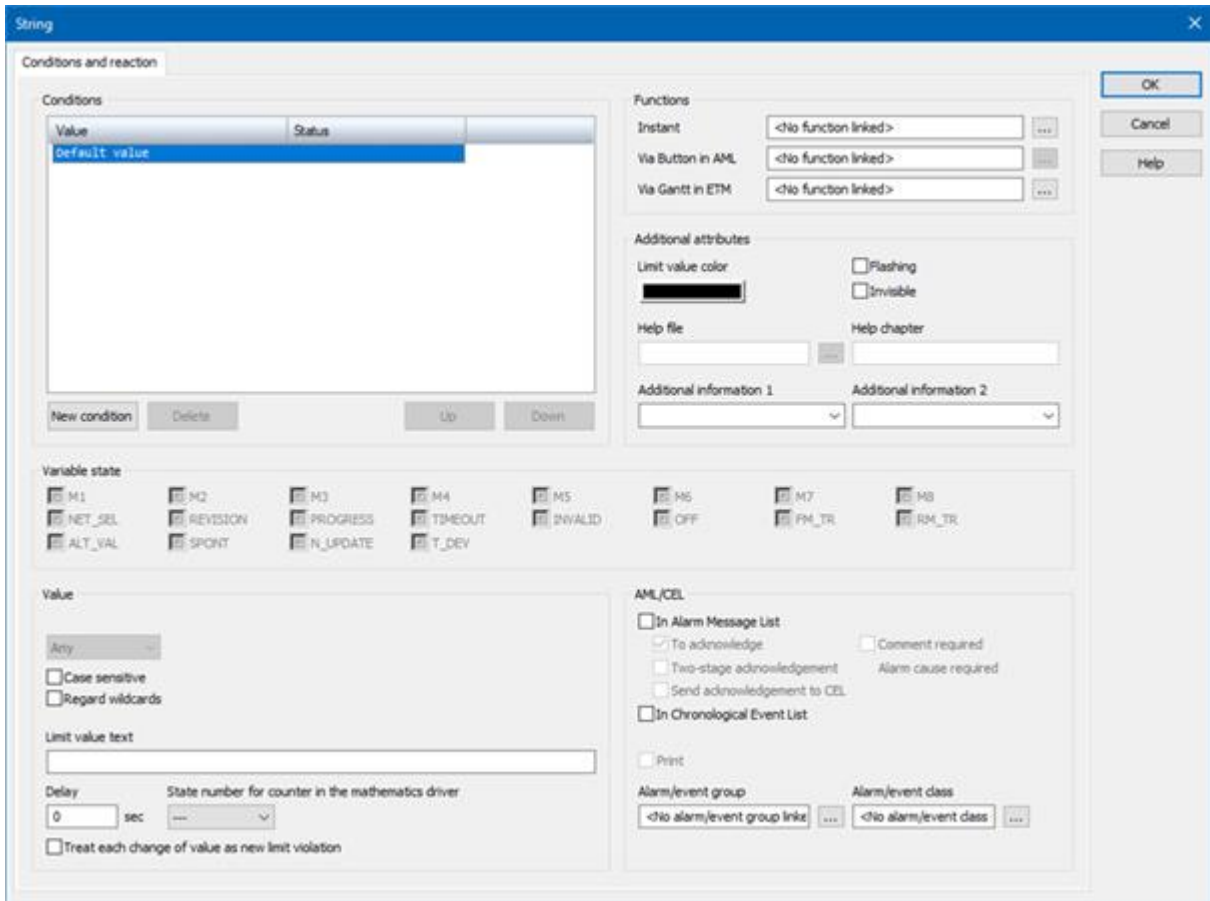In this tab special functions are configured.

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

Option	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

12.4.4 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.

STATES

Option	Description
Conditions list	List of the engineered states with value, state and cause of transmission.
New	Creates a new status based on the currently selected state.
Delete	Deletes all selected statues from the list.
Test	Opens the dialog for testing (on page 147) the condition.

Option	Description
Up	Moves selected states up.
Down	Moves selected states down.

VARIABLE STATE

Status bits selection.

Option	Description
State	Selection of the states via checkbox. Click in the checkbox in order to change the condition of the status. **Note:** For a list of all available status bits in zenon refer to manual Status processing chapter status bits. Not every status bit can be evaluated by means of reaction matrices.

VALUE

Option	Description
Value	Definition of the declaration of value. The value is entered in the input field next to the stipulated value given. Select from drop-down list: ▶ *any*: any change of value violates the limit value ▶ *greater*: Input of a limit value and a threshold value. The value to be triggered must be greater than the value selected. The following is applicable for *String* reaction matrices: The further back in the alphabet, the greater. Small letters are *greater* than capital letters. ▶ *smaller*: Input of a limit value and a threshold value. The value to be triggered must be less than the value selected. The following is applicable for *String* reaction matrices: The further forward in the alphabet, the smaller. Capital letters are *less* than small letters. ▶ *equal*:

Option	Description
	Entry of a limit value and a threshold value. The value to be triggered must be exactly the same value.
	▶ *Range*: Enter a range (from/to) and a threshold value.
	Note: Only values of numeric data types can be displayed in the **Value** column for AML and CEL. The column remains empty for entries with a **String** data type. However, **String** values can be displayed using a *dynamic text* (on page 178) element.
Match case	Evaluation takes capitalization into account in text.
	Note: Text is configured in the input field next to the stated value.
	▶ *Active:* Evaluation takes capitalization into account. The text entered must correspond to the prescribed value exactly in order for the status to be valid.
	▶ Inactive: Evaluation does not take capitalization into account.
	Note: Only available for *string* reaction matrices.
Take wildcards into account	Evaluation takes placeholders into account.
	Note: Wildcards are configured in the input field next to the stated value.
	▶ *Active:* Evaluation takes wildcards into account.
	▶ *Inactive*: Evaluation does not take wildcards into account. The values must correspond to the prescribed value exactly in order for a status to be valid.
	Wildcards:
	▶ *****: Replaces desired characters in the desired quantity. Can be used as a search term at any desired place. **red*** finds all texts that start with **red**.
	▶ **?**: Replaces precisely one character. **r?d** finds **red**, **rad**, ...

Option	Description
	Note: Only available for *string* reaction matrices.
Limit value text	Text which is displayed when at a limit value violation. Output in: ▸ dynamic text element ▸ CEL ▸ AML ▸ ...
Delay	Time period which the limit value violation must last in order for the limit value to be active.
State number for counter in the mathematics driver	Assignment of four possible status numbers for a counter in the mathematics driver. For this the variable must have a reaction matrix.
Treat each change of value as new limit violation	▸ *Active:* Each value change in the range of an engineered condition is displayed as separate violation (CEL, AML, ...).

EXAMPLE

The strings "*Mein,Name,ist,Hase*" are to be evaluated.
The evaluation is carried out with **value** *greater*.

Result:

▸ Sequence of evaluation: *ist,Name,Mein,Hase*.

Reason:

▸ *ist*: Starts with small letters. This is evaluated as greater than the following words with capital letters. It is therefore placed at the start.

▸ *Name*: N is greater than M and H. It is therefore placed at the second place.

▸ *Mein*: M is greater than H. It therefore placed at the third place.

▸ *Hase*: H is the lowest value It therefore placed at the last place.

FUNCTION

Option	Description
Immediately	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of

Option	Description
	button **No selection** in this dialog.
	The linked function is executed immediately if there is a limit value violation of the reaction matrix variable.
With button in AML	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog.
	Only available if you activated option **In Alarm Message List** in **AML/CEL**.
	The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the corresponding entry has been selected in the **Alarm Message List (AML)** and the **Execute Function** button is clicked on.
With button in Gantt	Click on **...** to open the dialog to link a function. You can deselect an already linked function with the help of button **No selection** in this dialog.
	The linked function is executed in the event of a limit value violation of the reaction matrix variable as soon as the Gantt curve is clicked on with the mouse pointer in the area of the limit value violation in the Runtime in the **Extended Trend (ETM)**.

ADDITIONAL ATTRIBUTES

Option	Description
Limit value color	Color when a limit value has been violated.
Flashing	▸ *Active*: Flashes when a limit value has been violated.
Invisible	▸ *Active*: Will be switch to invisible when a limit value 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**.
	The linked help file is opened in the Runtime if:
	▸ A help chapter has been entered

Option	Description
	▸ The corresponding alarm is selected in the Alarm Message list and ▸ The **Open 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. Must contain an entry in order for help to be opened in the Runtime. **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

Option	Description
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.
Two-stage acknowledgment	Only available if **In Alarm Message List** is activated. ▸ *Active*: **Two-stage acknowledgement** is required.
Alarm cause required	Only available if **In Alarm Message List** is activated. ▸ *Active*: **Alarm cause** is required
In Chronological Event List	▸ *Active*: Will be entered in the CEL. **Note**: 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 value violation has been rectified and then is violated

Option	Description
	again, or the state becomes inactive and then active again, is a CEL entry generated.
Print	Only available if **In Alarm Message List** or in **In Chronological Event List** is activated. ▸ *Active*: Will be printed via the set standard printer.
Alarm/Event Group	Allocation to an alarm/event group. Select from drop-down list.
Alarm/Event Class	Allocation to an alarm/event class. Allocation to an alarm/event group. Select from drop-down list.

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

12.5 Dynamic limit value texts in reaction matrices

Dynamic limit value texts can also be displayed in reaction matrices:

$ %PV
$ @text

See also Dynamic limit value text (on page 125).

Reaction matrices can be used for multiple variables.

12.6 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:** active

▶ **Delay:** *>0*
The delay time is calculated from the time stamp of the variables.

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.

12.7 List of status bits

Bit number	Short term	Long name	zenon Logic identifier
0	M1	User status 1; for Command Processing: Action type "Block"; Service Tracking of the IEC 850 driver	_VSB_ST_M1
1	M2	User status 2	_VSB_ST_M2
2	M3	User status 3	_VSB_ST_M3
3	M4	User status 4	_VSB_ST_M4
4	M5	User status 5	_VSB_ST_M5
5	M6	User status 6	_VSB_ST_M6
6	M7	User status 7	_VSB_ST_M7
7	M8	User status 8	_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	Command "Timeout exceeded" (command runtime exceeded)	_VSB_RTE
12	MAN_VAL	Manual value	_VSB_MVALUE
13	M14	User status 14	_VSB_ST_14
14	M15	User status 15	_VSB_ST_15
15	M16	User status 16	_VSB_ST_16

Bit number	Short term	Long name	zenon Logic identifier
16	GI	General interrogation	_VSB_GR
17	SPONT	Spontaneous	_VSB_SPONT
18	INVALID	Invalid	_VSB_I_BIT
19	T_STD_E	External standard time (standard time) **Caution:** up to version 7.50, this was the status bit T_CHG_A	_VSB_SUWI
20	OFF	Switched off	_VSB_N_UPD
21	T_EXTERN	Real time - external time stamp	_VSB_RT_E
22	T_INTERN	Internal time stamp	_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	_VSB_AVALUE
28	RES28	Reserved for internal use (alarm flashing)	_VSB_RES28
29	N_UPDATE	Not updated (zenon network)	_VSB_ACTUAL
30	T_STD	Internal 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

Bit number	Short term	Long name	zenon Logic identifier
36	COT4	Cause of transmission bit 5	_VSB_TCB4
37	COT5	Cause of transmission bit 6	_VSB_TCB5
38	N_CONF	Negative confirmation of command by device (IEC 60870 [P/N])	_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	Default 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	External time stamp invalid	not defined
50	CB_TRIP	Breaker tripping detected	not defined
51	CB_TR_I	Breaker tripping detection inactive	not defined
52	OR_DRV	Value out of the valid range (IEC 61850)	not defined
53	T_UNSYNC	ClockNotSynchronized (IEC 61850)	not defined
54	PR_NR	Not recorded in the Process Recorder	not defined
55	T_DEV	Configured time difference between internal and external timestamp reached.	not defined

Bit number	Short term	Long name	zenon Logic identifier
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

☀ Information

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

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

13 Flashing

Screen elements and entries in the **Alarm Message List** or the **Chronological Event List** can be displayed in a flashing way in the Runtime,

13.1 Flashing - General

Screen elements and entries in the **Alarm Message List** or the **Chronological Event List** can be displayed in a flashing way in the Runtime,

You need to carry out the following configuration for the variable and the Screen element:

▶ Variable or reaction matrix:

   ▶ Engineering at variable
   Property **Flashing** in segment **Additional attributes** in property group **Limit Values**.

> ▶ Project configuration in a reaction matrix
> When configuring the parameters of a reaction matrix in the **States and Reactions** configuration dialog by activating the **Flash** option.
> **Info:** This option can be found in section **Additional attributes** of the REMA configuration dialog.

▶ Display in the Runtime:

> ▶ Screen elements:
> Section **Flashing** in Screen element property group **Visibility/flashing**.

> ▶ Alarm Message List:
> In order to display unacknowledged alarms in a flashing way in the Alarm Message List enable project property **Unacknowledged alarms flash**.
> **Info:** You can find this property in project property group **Alarm Message List**.
> With function **Alarms: acknowledge flashing** you can deactivate the flashing in the Runtime.
> The flashing behavior is inherited from the variable engineering.

13.2 Engineering in the Editor

Screen elements are flashing at visualization in zenon. In order to do this the following engineering steps are necessary:

▶ Configure the relevant project properties for the visualization of flashing:

> ▶ **Flash freq. [tenth sec]** In the project property group **Graphical design**.

> ▶ **Unacknowledged alarms flash** In the project property group **Alarm Message List**

▶ Configure a variable:

▶ Configure one or several limits for a variable.

> ▶ Enable property **Flashing** for your configured limits.

▶ Configure a Screen element for a zenon Screen.

> ▶ Configure the flashing behavior for the display in the zenon Runtime at the Screen element in property group **Flashing**.

13.3 Display unacknowledged alarms as flashing

Unacknowledged alarms can be displayed as flashing in the AML.

With the **Unacknowledged alarms flash** property active, the foreground color and background color alternate in all unacknowledged and unselected lines in the Alarm Message List every second. Selected lines do not flash.

To do this:

1. In the Editor, navigate to the **Alarm Message List** section in Project Properties.

2. Activate the checkbox in front of the **Unacknowledged alarms flash** property.

3. Configure the desired colors (foreground) for **Alarm received**, **Alarm cleared**, **Alarm acknowledged**

   Hint: Set the **Alarm/event class color** property to *as line background*. The color of the alarm class is then used as a background color.

13.4 Display in the zenon Runtime.

The following is true for flashing in the zenon Runtime:

▶ Screen elements are flashing:

  ▶ for limit violations.

    when the limit was violated once and an alarm which must be acknowledged was engineered for this limit.
    In this case the Screen element continues to flash until the alarm was acknowledged or flash acknowledged. For more information on flash acknowledgment refer to the following section. The Screen element will also continue to flash if the limit violation was removed due to a value change.
    Engineering info: The need for acknowledgment is enabled via limit property **To acknowledge**.

▶ The flashing of Screen elements with alarms which must be acknowledged is canceled:

  ▶ by flash acknowledgment via Double-click

  ▶ by flash acknowledgment via function **Alarms: acknowledge flashing**

  ▶ by acknowledging the alarm in the Alarm Message List

  ▶ by alarm acknowledgment via function **Acknowledge alarms**

> **The flashing is also canceled if the limit violation is removed by value change and the limit violation was not engineered as must be acknowledged.**⚠**Attention**
>
> The **Alarms: acknowledge flashing** function must not be mistaken for **Acknowledge alarms**.
>
> ▸ Function **Alarms: acknowledge flashing**: Only ends the flashing, does not acknowledge the alarm.
>
> ▸ Function **Acknowledge alarms**: Acknowledges the alarm and also ends the flashing.
>
> If the acknowledgment of the flashing should also acknowledge the alarm, the **Flashing acknowledgement** property must also be activated in the Editor in the in project properties in the **Alarm Message List** group. In this case, the alarm itself is also acknowledged when acknowledging the flashing in the AML.

13.5 Flashing in the zenon network

If a screen element flashes in a zenon screen or an alarm in the Alarm Message List, this display is adopted in the Runtime by all clients and servers in the zenon network and displayed on all computers the same. This assures that in the whole network on all computers the current flashing state is always displayed identically.

14 Functions for variables

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

To create a function:

1. select **New function**... in the context menu or in the toolbar
2. navigate to the **variable**
3. select the desired function

14.1 Export data

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

The export format and the file name are defined in the Editor. The save location when executed in the Runtime is the *Export* folder in the Runtime folder.

Hint: 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 185).

2. Select **Export data**.
   The configuration dialog is opened.
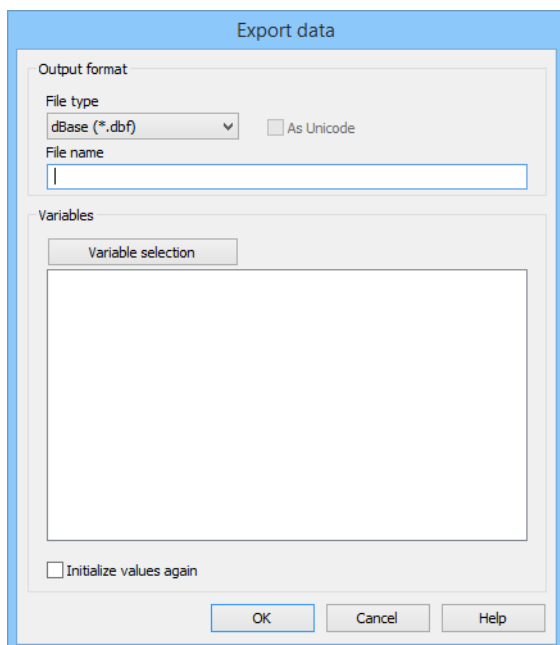
OUTPUT FORMAT

Parameter	Description
File type	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).

Parameter	Description
File names	Create the filenames. A maximum of 8 characters are permitted for DBF files. **Note:** The name can no longer be changed in the Runtime. Save location in the Runtime: *Export* folder in Runtime folder.

VARIABLES

Parameter	Description
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 the 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:

14.2 Read dBase file

This function is used to read in a   DBase file during online operation and set its content as set values.

> 💡 **Information**
>
> This function is only for compatibility purposes.
> **Recommendation:** Instead, use zenon Logic to write individual values or
> recipes and the Recipe Group Manager for block-wise writing.

Names of variables and the attendant set values are contained in the DBase file. 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.

> ▸ Transfer parameters: Name of the **DBase** file.
> Name and folder name must correspond to DOS conventions.

The dBaseIV file must have the following structure and contents:

Column titles	Type	Value	Comment
NAME	*Char*	*32*	Name of the variable.
WERT	*Num*	*20*	Value of the variables.
TYP	*Char*	*8*	Data type. Possible values: ▸ **C***: String* ▸ **N***: Numeric*
DIRECTION	*Char*	*12*	(is not currently used)

Attention: The column width must correspond to the given values exactly. You should therefore not edit the file in a table calculation, because the column width is changed in the process. Use a program for creation that can transfer fixed column widths and Headers.

14.3 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 a dialog.
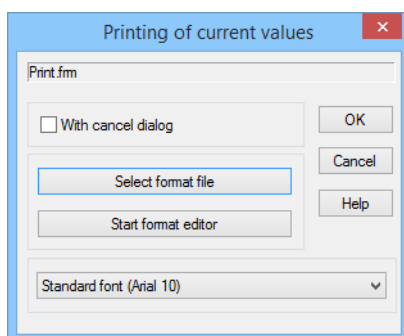
Configurable options are:

Parameter	Description
With cancel dialog	displays cancel 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

Parameter	Description
Font	sets font which will be used

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

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

> ⚠ **Attention**
>
> The QRF definition format expected by the code is, for example:
> *define "$001" as "REAL_VAR_NAME"*
> The quotes are important for this!

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	Condition 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

14.4 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.

14.5 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.

14.6 HD administration inactive/active

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

14.7 Write set value

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 a dialog. The function is configured with its own dialogs for:

▸ numerical variables (on page 193)

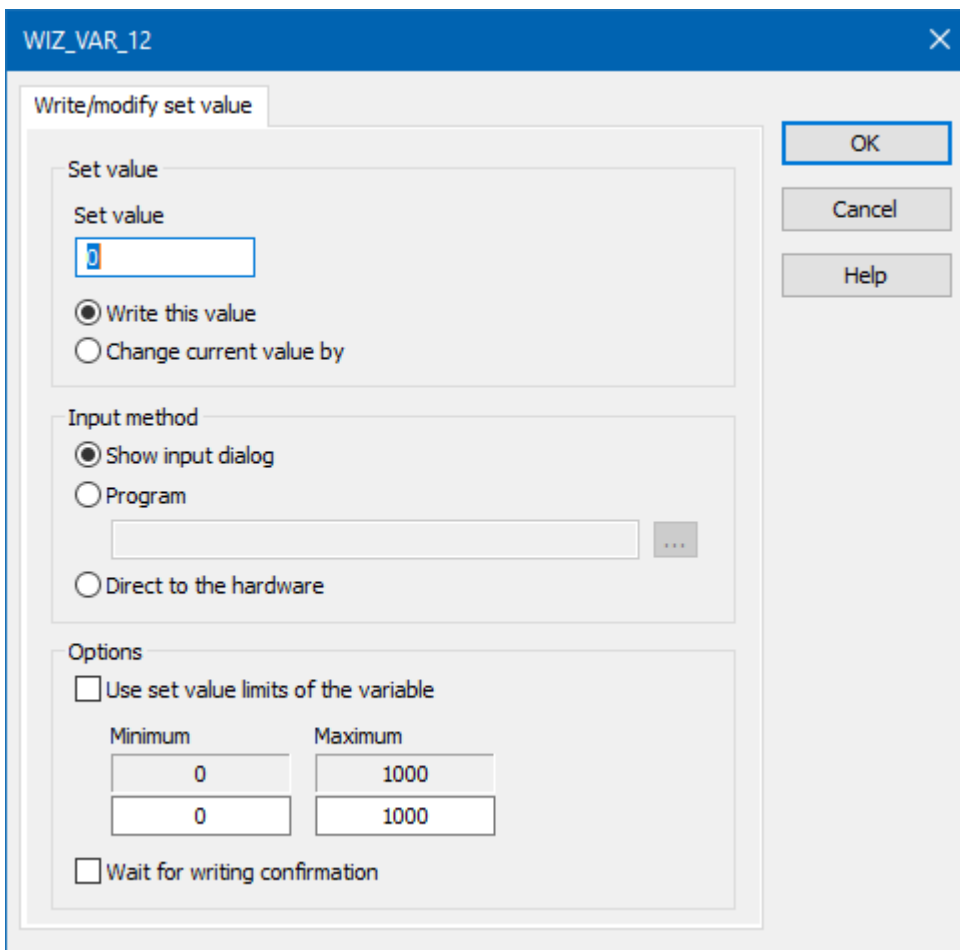▸ binary variables (on page 196)

▸ string variables (on page 198)

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

💡 Information

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

14.7.1 Write set value/modify for numeric variables

SET VALUE

Parameter	Description
Set value	Input of the set value to be set for the variable.
Write this value	The value entered under **set value** is written.
Change current value by	The current value is amended by the value entered under **Change by**. **Note:** Change by is displayed as soon as **Change current value by** has been selected. **Example:** In a screen, you can configure a button that increases the current value in the Runtime by 10. To do this, go to **Change current value by** and enter 10 into the text field.

INPUT METHOD

Parameter	Description
Show input dialog	A set value is entered using a specific dialog.
Program	A set value is entered via a program. Select **Program** and click on the **...** button to open the selection dialog. Select the desired file in ***.EXE** format and confirm this selection by clicking **Open**.
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 writing the set value, an input dialog is opened and the user can change the settings in the Runtime.

OPTIONS

Parameter	Description
Use set value limits of the variable	*Active*: The set value limits are taken from the variable definition (**Description/Set value**). *Inactive*: The permitted set value limits can entered in the function directly (**minimum** and **maximum**).
Minimum	Input of the minimum value.

Parameter	Description
Maximum	Input of the maximum value.
Wait for writing confirmation	*Active*: The function only finishes 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 ensure that the following function is only started once this one is actually finished. The confirmation can also be evaluated with the status bit **WR-SUC**. *Inactive*: Do not wait for the writing confirmation.

CLOSE DIALOG

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.

💡 Information

A decimal value can be entered with both a comma (**,**) as well as with a period (**.**). The decimal separator is automatically changed to a period internally.

14.7.2 Write set value/modify for binary variables

SET VALUE

Option	Description
Write predefined value	A value that has been defined in advance has been written: ▶ **Off (0)**: Value of variables will be set to *Off (0)*. If there is a limit value text, this is also shown. ▶ **On (1)**: Value of variables will be set to *ON (1)*. If there is a limit value text, this is also shown. ▶ **Toggle**: Value of the variables is switched to complementary (opposite) status. **Tip:** If the variable value is to be toggled, we recommend activating the **Permanently read variable** 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.

Option	Description
Propose current value	*Active*: The value is read from the hardware and then proposed as a default or sent directly to the hardware. *Inactive* : The value defined under **Set value** is proposed as a default or sent directly to the hardware.

INPUT METHOD

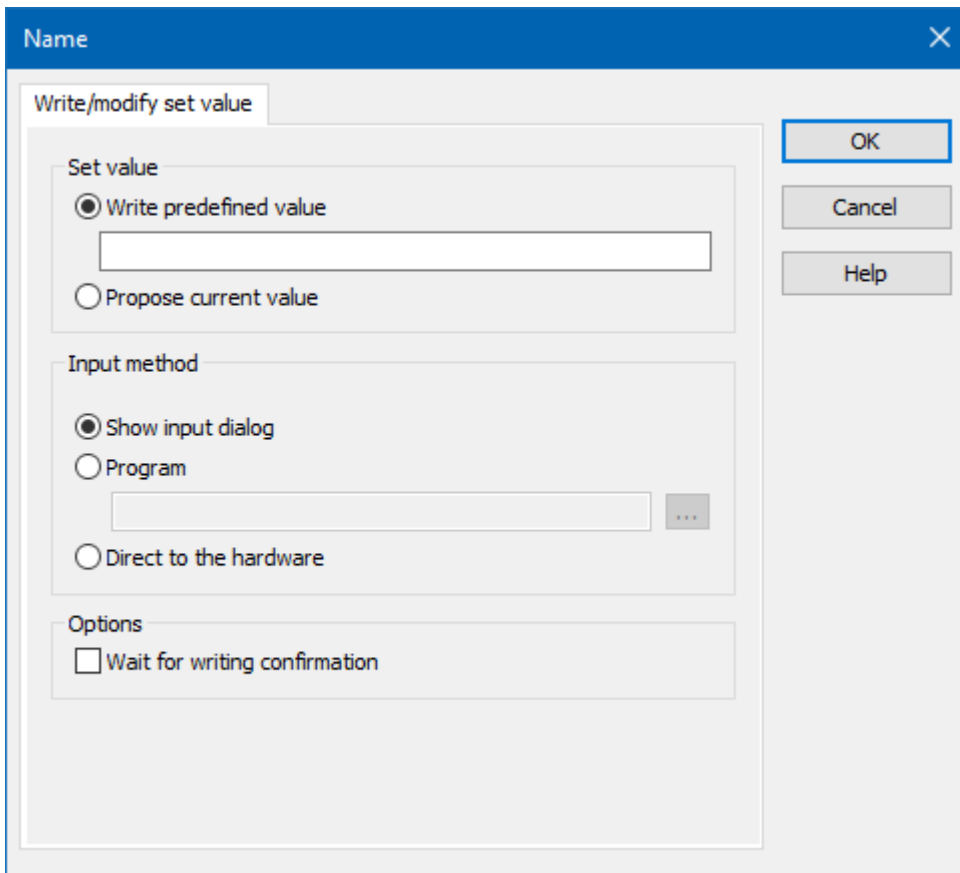| Option	Description
Show input dialog	A set value is entered using a specific dialog.
Program	A set value is entered via a program. Select **Program** and click on the **...** button to open the selection dialog. Select the desired file in ***.EXE** format and confirm this selection by clicking **Open**.
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 writing the set value, an input dialog is opened and the user can change the settings in the Runtime.

OPTIONS

Option	Description
Wait for writing confirmation	*Active*: The function only finishes execution once a positive confirmation of the write action is received or the fixed defined timeout of 30 seconds is over. It is thus possible, in scripts, to ensure that the following function is only executed once this has actually been completed. The confirmation can also be evaluated with the status bit **WR-SUC**. *Inactive*: Do not wait for the writing confirmation.

CLOSE DIALOG

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.

14.7.3 Write set value/modify for string variables

Note: No configuration can be carried out for write-protected variables.

SET VALUE

Parameter	Description
Write predefined value	A value that has been defined in advance is written:
Propose current value	*Active*: The value currently in the controller is proposed as a default and sent directly to the hardware.
	Inactive: The value defined under **Set value** is proposed as a default or sent directly to the hardware.

INPUT METHOD

Parameter	Description
Show input dialog	A set value is entered using a specific dialog.
Program	A set value is entered via a program.

Parameter	Description
	Select **Program** and click on the **...** button to open the selection dialog. Select the desired file in ***.EXE** format and confirm this selection by clicking **Open**.
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 writing the set value, an input dialog is opened and the user can change the settings in the Runtime.

OPTIONS

Parameter	Description
Wait for writing confirmation	*Active*: The function only finishes 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 ensure that the following function is only started once this one is actually finished. The confirmation can also be evaluated with the status bit **WR-SUC**. *Inactive*: Do not wait for the writing confirmation.

CLOSE DIALOG

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.

14.7.4 Check write set value

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

▸ **WR-ACK**

The driver received a value for writing.

▸ <u>**WR-SUC**</u>

Value *1*: Writing successful.

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

> ▸ 💡 Information
>
> 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 192): Activate option **Wait for writing confirmation** in the configuration dialog of the function.

▸ **Standard recipes**: Activate property **Write synchronously**.

▸ **Recipe Group Manager** Activate property **Write synchronously**.

ENTRY IN CEL

You can find the following properties in the project properties under **Chronological Event List**:

▸ **Function Write set value**

For the entry in the CEL you must activate property **Function Write set value** in node **Chronological Event List** in the project settings. After this the positive or negative response the execution of the function is written to the CEL.

‣ **Log set value for set value changes only**

If the function is activated, there is no entry in the CEL if the same value is placed again or received. Only value changes are logged. The execution of the property does not depend on the **Old and new value** variable setting.

‣ **Text on set value change**

Allows the creation of free text for the display of the new value. You can find information on the necessary requirements and the use of placeholders in the Use of placeholders for changes to a set value.

‣ **Text on set value change (old/new value)**

Allows the creation of free text for the display of the old value and the new value. You can find information on the necessary requirements and the use of placeholders in the Use of placeholders for changes to a set value.

If a recipe or a recipe group is saved again or duplicated under another name, the process is depicted in the corresponding CEL entry.

The following will be displayed in the CEL entry:

‣ The name of the original recipe or the recipe group

‣ The name of the new recipe or the recipe group

‣ The Recipe version

Note: This information is relevant when a recipe or recipe group has been copied and saved again. Changes can easily be retraced via the CEL entry.

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 value or reaction matrix (on page 131). The query is carried out via a multi analog (on page 162) or a multi binary (on page 157) reaction matrix.

14.8 Driver command function

The zenon **Driver commands** function is to influence drivers using zenon.
You can do the following with a driver command:

‣ Start

‣ Stop

‣ Shift a certain driver mode

‣ Instigate certain actions

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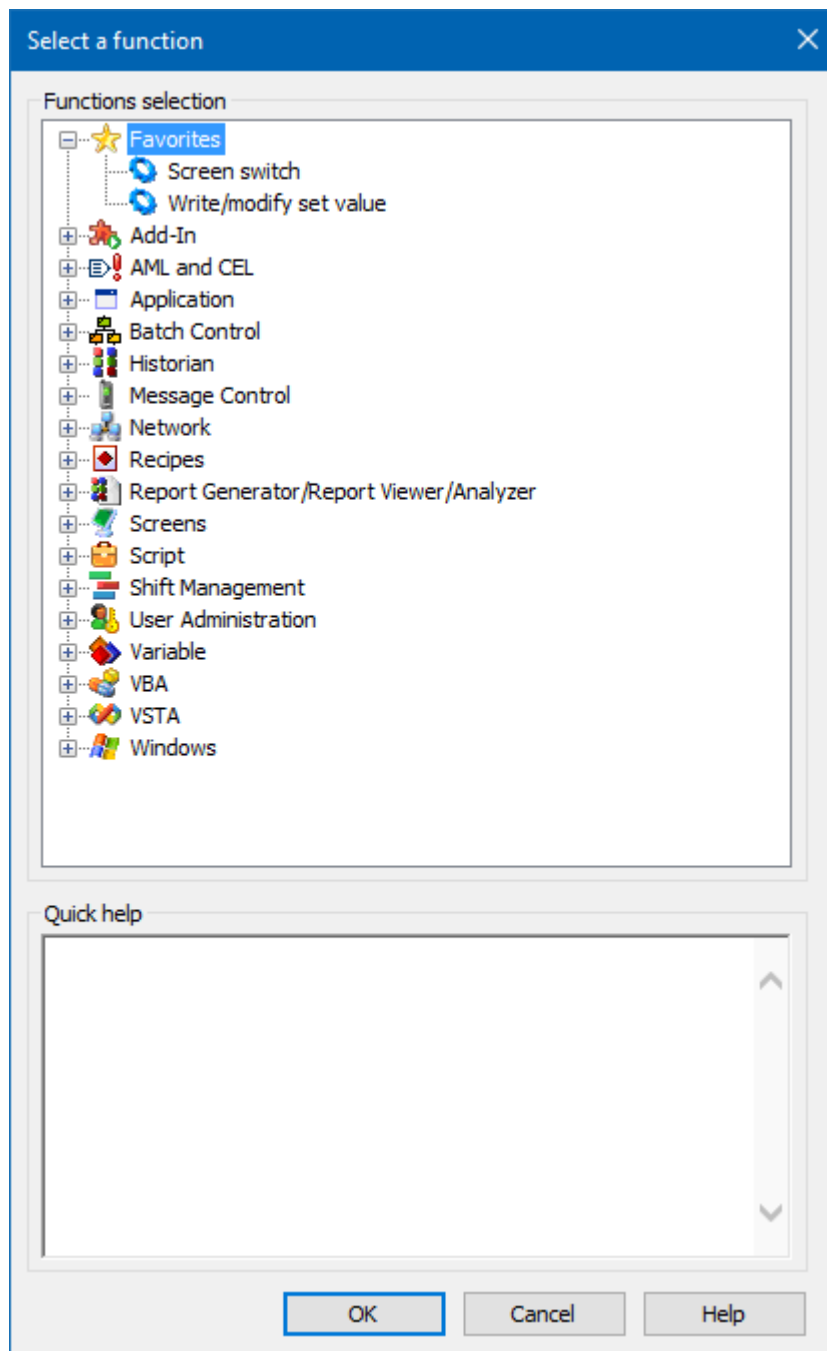
> **⚠Attention**
>
> The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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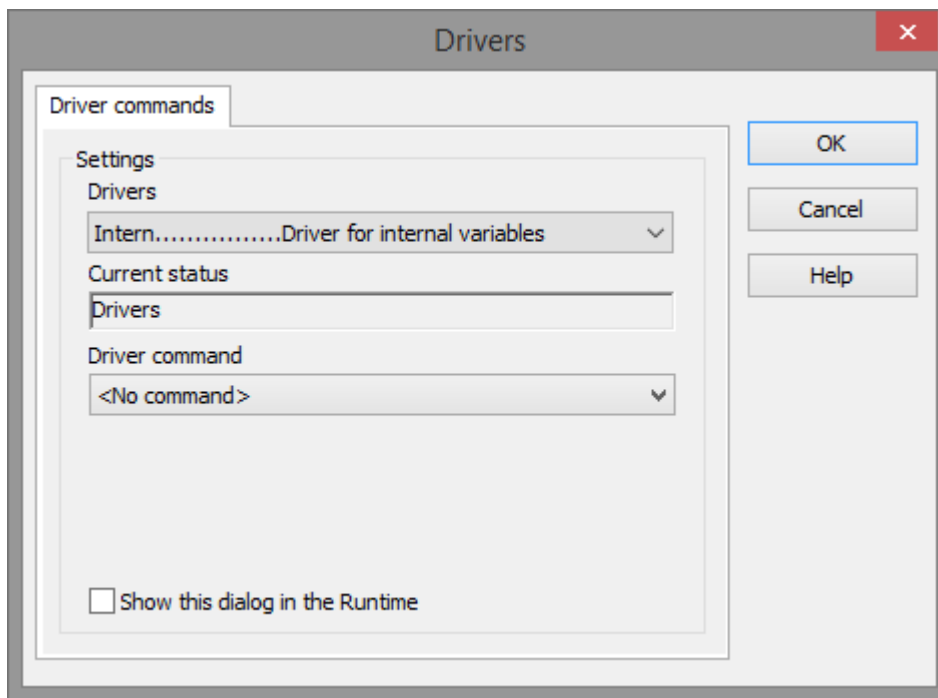To configure the function:

1. Create a new function in the zenon Editor.

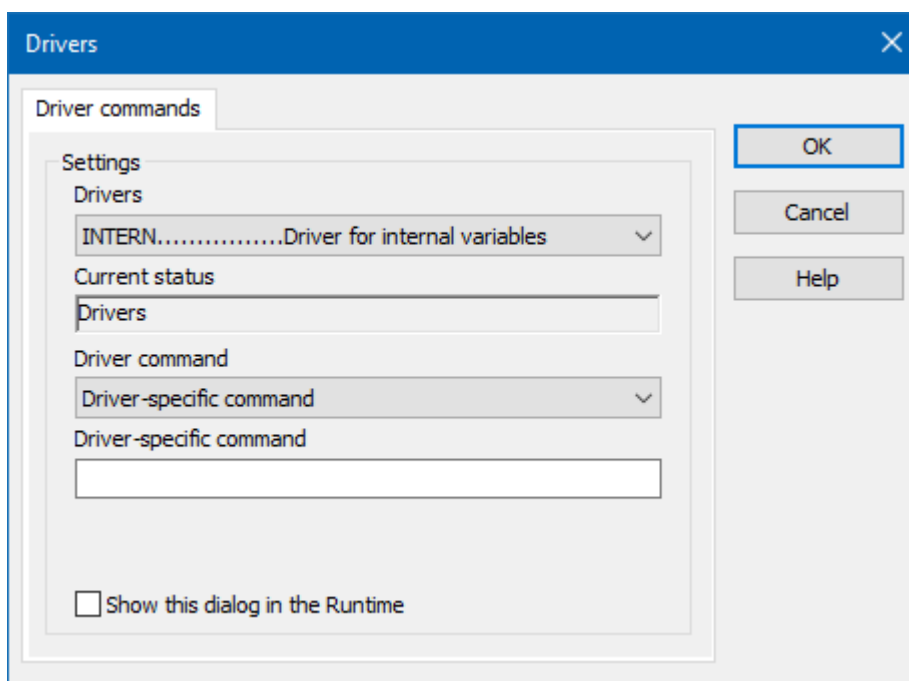The dialog for selecting a function is opened

2. Navigate to the node **Variable.**
3. Select the **Driver commands** entry.

The dialog for configuration is opened

4. Select the desired driver and the required command.

5. Close the dialog by clicking on **OK** and ensure that the function is executed in the Runtime. Heed the notices in the **Driver command function in the network** section.

DRIVER COMMAND DIALOG

Option	Description
Driver	Selection of the driver from the drop-down list. It contains all drivers loaded in the project.
Current condition	Fixed entry that is set by the system. no function in the current version.
Driver command	no function in the current version. For details on the configurable driver commands, see the **available driver commands** section.
Driver-specific command	Entry of a command specific to the selected driver. **Note:** Only available if, for the **driver command** option, the *driver-specific command* has been selected.
Show this dialog in the Runtime	Configuration of whether the configuration can be changed in the Runtime: ▸ *Active*: This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution. ▸ *Inactive*: The Editor configuration is applied in the Runtime when executing the function. Default: *inactive*

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

AVAILABLE DRIVER COMMANDS

These driver commands are available - depending on the selected driver:

Driver command	Description
No command	No command is sent. A command that already exists can thus be removed from a configured function.

| Driver command	Description
Start driver (online mode)	Driver is reinitialized and started. **Note:** If the driver has already been started, it must be stopped. Only then can the driver be re-initialized 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	Entry of a driver-specific command. Opens input field in order to enter a command.
Driver - activate set setpoint value	Write set value to a driver is possible.
Driver - deactivate set setpoint value	Write set value to a driver is prohibited.
Establish connecton with modem	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
Disconnect from modem	Terminate connection (for modem drivers)
Driver in counting simulation mode	Driver is set into counting simulation mode. All values are initialized with *0* and incremented in the set update time by *1* each time up to the maximum value and then start at *0* again.
Driver in static simulation mode	No communication to the controller is established. All values are initialized with *0*.
Driver in programmed simulation mode	The values are calculated by a freely-programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in the zenon Logic Runtime.

DRIVER COMMAND FUNCTION IN THE NETWORK

If the computer on which the **Driver commands** function is executed is part of the zenon network, further actions are also carried out:

▶ A special network command is sent from the computer to the project server.
It 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.

14.9 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
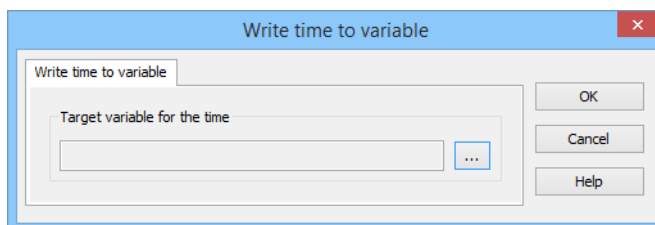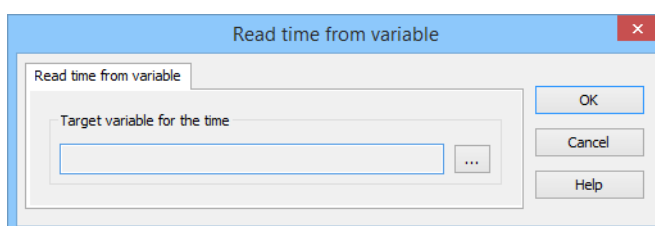
The name of the variable has to be given as a parameter.

14.10 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*.
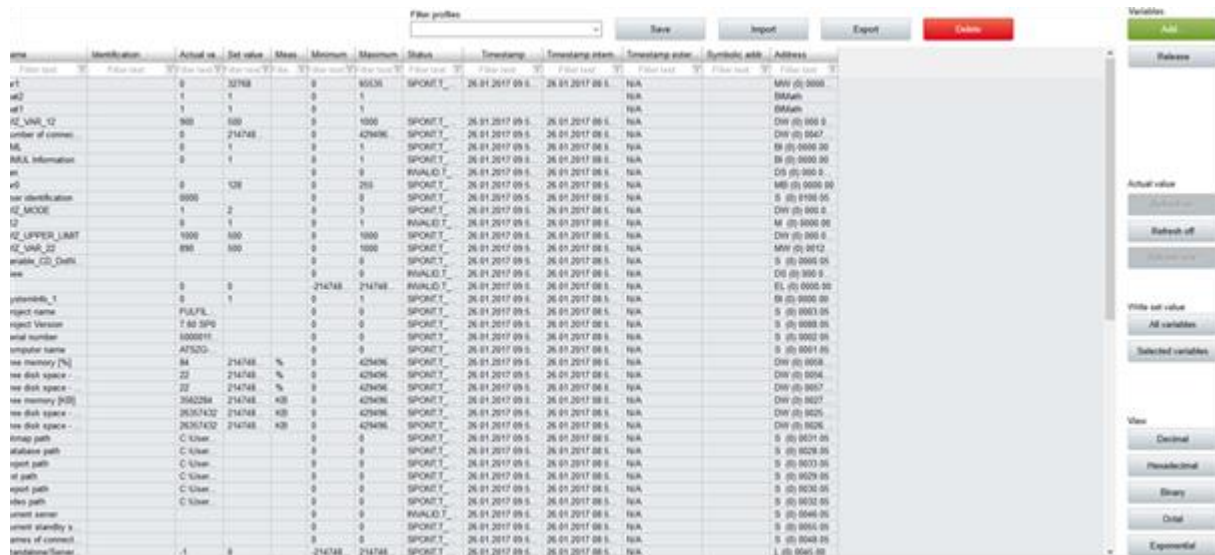
Note: Time is saved as local time. For details see chapter Handling of date and time in chapter Runtime.

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

15 Screen variable-diagnosis

With the screen type **variable-diagnosis** it is possible to display variables in the runtime and to set target values.

> 💡 **Information**
>
> You can find general information on the pre-defined screen types in the Screens manual in the screen types chapter.

15.1 Create screen Variable diagnosis

ENGINEERING

Two procedures are available to create a screen:

- ▶ The use of the screen creation dialog
- ▶ The creation of a screen using the properties

Steps to create the screen using the properties if the screen creation dialog has been deactivated in the menu bar under **Tools**, **Settings** and **Use assistant**:

1. Create a new screen.

   To do this, select the **New screen** command in the tool bar or in the context menu of the **Screens** node.

2. Change the properties of the screen:

   a) Name the screen in the **Name** property.

b) Select *Variable diagnosis* in the **Screen type** property.

c) Select the desired frame in the **Frame** property.

3. Configure the content of the screen:

a) Select the **Elements (screen type)** menu item from the menu bar.

b) Select *Insert template* in the drop-down list.
The dialog to select pre-defined layouts is opened. Certain control elements are inserted into the screen at predefined positions.

c) Remove elements that are not required from the screen.

d) If necessary, select additional elements in the **Elements** drop-down list. Place these at the desired position in the screen.

4. Create a screen switch function.
Configure this screen switching function.
You can find further information about this in the Screen switching - variable diagnosis (on page 216) chapter.

15.2 Create screen Variable diagnosis

The variable diagnosis screen visualizes variables in the Runtime

Control element	Description
Insert template...	Opens the dialog for selecting a template for the

Control element	Description
	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 position 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 placed in the zenon screen. Elements can be moved on the screen and arranged individually.

VARIABLE LIST

List window in the Runtime:

▶ The columns can be sorted and filtered in Runtime.

▶ Filtering and sorting can be saved in the filter profiles.

▶ The column width can be amended by holding down the right mouse button.

▶ Columns can be rearranged by means of drag&drop.

Column	Description
Name	Variable name. This corresponds to the **Name** property in the zenon Editor. Default: *Visible in Runtime* **Note:** cannot be changed in Runtime
Identification	Identification of the variable. This corresponds to the **Identification** property in the zenon Editor. Default: *Visible in Runtime* **Note:** cannot be changed in Runtime
Actual value	Current value of the variable. Updating by means of **Updating on** or **Update now** buttons.

Column	Description
	Default: *Visible in Runtime* **Note:** cannot be changed in Runtime
Set value	Set value of the variable. Default: *Visible in Runtime*
Measuring unit	Measuring unit of the variables. This corresponds to the **Measuring unit** property in the zenon Editor. Default: *Visible in Runtime* **Note:** cannot be changed in Runtime
Minimum	Minimum signal range of the variables in the PLC. This corresponds to the **Signal range min.** property in the zenon Editor. Default: *Visible in Runtime* **Note:** cannot be changed in Runtime
Maximum	Minimum signal range of the variables in the PLC. This corresponds to the **Signal range max.** property in the zenon Editor. Default: *Visible in Runtime* **Note:** cannot be changed in Runtime
State	Current status of the variable in plain text display. Default: *Visible in Runtime* **Note:** cannot be changed in Runtime
Timestamp	Current time stamp of the variable. ▸ If a value has the status *T_EXTERN* and no *T_INVAL*, the value of the the external time stamp is used as a time stamp. The external time stamp is the time value that the controller has provided together with the value at the driver.

Column	Description
	If a value has the status *T_INTERN* or has *T_EXTERN* and *T_INVAL,*, the value of the internal time stamp is used as a time stamp. The internal time stamp is the time point at which the driver has received the value from the controller.Display with precision in seconds.
	Default: *Visible in Runtime*
	Note: cannot be changed in Runtime
Timestamp internal (UTC)	Current internal time stamp of the variable in UTC time.
	Display in microsecond resolution.
	Default: *Visible in Runtime*
	Note: cannot be changed in Runtime
Timestamp external (UTC)	Current external time stamp of the variable in UTC time.
	Display in microsecond resolution.
	Default: *Visible in Runtime*
	Note: cannot be changed in Runtime
Symbolic address	Symbolic address of the variable.
	This corresponds to the **Symbolic address** property in the zenon Editor.
	Empty, if not present.
	Default: *Visible in Runtime*
	Note: cannot be changed in Runtime
Address	Address of the variable
	This corresponds to the **Addressing** property group in the zenon Editor.
	Default: *Visible in Runtime*
	Note: cannot be changed in Runtime

Column	Description
Actual value (raw data)	Raw data of the variable's actual value. This value corresponds to the actual value of the variable before the value adjustment. Default: *Is not shown in Runtime by default.* **Note:** cannot be changed in Runtime
Status (binary)	Status of variable. Display format: shows the complete 64-bit status register in binary form. This display is always binary, regardless of the selected value display in the display options. Default: *Is not shown in Runtime by default.* **Note:** cannot be changed in Runtime
Timestamp external (local)	Current external time stamp of the variable in local time. Display in microsecond resolution. N/A if there is no external time stamp. Default: *Is not shown in Runtime by default.* **Note:** cannot be changed in Runtime
Timestamp internal (local)	Internal time stamp of the variable in local time. Display in microsecond resolution. Default: *Is not shown in Runtime by default.* **Note:** cannot be changed in Runtime

DISPLAY OPTIONS

Control elements for the display of counter values of the variable list. Switching of the value display resets the column width back to the default. For example, manually-widened columns become wider as a result.

In the variable lists, the following content is visualized in counter values and can be switched using the value display:

▸ Actual value

- ▶ Set value
- ▶ Minimum
- ▶ Maximum
- ▶ Actual value (raw data)

Control element	Description
Value display: Decimal	Variable list values are shown in decimal form.
Value display: Hexadecimal	Variable list values are shown in hexadecimal form.
Value display: Binary	Variable list values are shown in binary form.
Value display: Octal	Variable list values are shown in octal form.
Value display: Exponential	Variable list values are shown in decimal exponential form.

VARIABLES

Control element	Description
Add variable...	Add a variable to the variable list in the Runtime
Remove variable	Remove variable from the variable list in the Runtime

UPDATE VALUE

Control element	Description
Refresh actual value on	Switch on automatic updating of current variable values.
Refresh 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.

SELECTED VARIABLES

Control element	Description
Switch to spontaneous value	Switch selected variable to spontaneous value.
Switch to alternate value	Switch selected variable to alternate value.
Modify alternate value	Modify the alternate value of the selected variable.
Set revision	Switch on the revision for selected variables. To do this, the *REVISION* status bit (bit *09*) is set.
Reset revision	Switch off the revision for selected variables. To do this, the *REVISION* status bit (bit *09*) is reset.
Write state OFF	Switch on the spontaneous value for the selected variable. To do this, the *OFF* status bit (bit *20*) is reset.
Reset state OFF	Switch off the spontaneous value for the selected variable. To do this, the *OFF* status bit (bit *20*) is set.

FILTER PROFILES

Buttons for filter settings in the Runtime.

Control element	Description
Profile selection	Select profile from list. Select from drop-down list: This drop-down list contains profiles that have been saved with the **Save** button.
Save	Saves current setting as a profile. **Note:** The name can be a maximum of 31 characters long and must only contain valid characters. Prohibited are: **! \ / : * ? < > | ""** Saved profiles are shown in the **Profile selection** drop-down list. **Note:** The name can be a maximum of 31 characters long and must only contain valid characters. Prohibited are: **! \ / : * ? < > | ""**

Control element	Description
Delete	Deletes selected profile. The deleted profile is no longer available in **Profile selection**.
Import	Imports filter profiles from export file. Imported filter profiles are then available in **Profile selection**.
Export	Exports filter profiles in the file.

💡 Information

A decimal value can be entered with a **comma** as well as with a **period** as a separator. The decimal separator is automatically changed to a decimal **point**.

👍 Hint

You can filter for the NORM and N_NORM status bits.

# 15.3 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**:

1. Select **New function...**.

2. Select **screen switching** and confirm this selection by clicking on **OK**.

   The **screen selection** dialog is opened.

3. Select the variable diagnosis screen (on page 208)

4. Confirm the selection by clicking on **OK**.

The **Filter** dialog is opened.
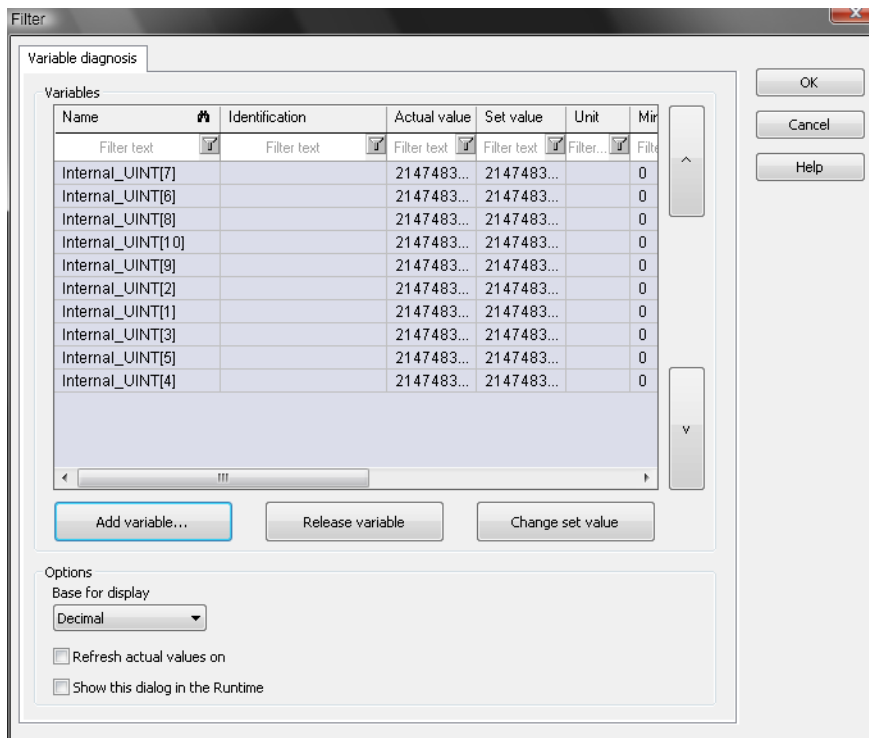
Parameter	Description
Variables	List of the variables and their properties.
Add variable	Opens the dialog for the variable selection in order to add one or more variables.
Release variable	Selected variables are released from the screen.
Change set value	The default set values for the variables of the screen can be changed. Set values can also be changed by clicking the cell with the set value.

OPTIONS

Parameter	Description
Base for display	Defines the display of the values in the Runtime. Selection of the value change from a drop-down list: Possible values: ▸ Decimal ▸ Hexadecimal ▸ Binary ▸ Octal

Parameter	Description
	▸ 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	Selection of whether the dialog is shown in Runtime. ▸ *Active*: Dialog is displayed in Runtime Settings can be edited before execution. ▸ *Inactive*: Dialog is not offered in Runtime. The settings configured in the Editor are applied.

💡 Information

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.

You can copy individual cell content from the variable list in the Runtime:

1. Go to the desired entry with the mouse pointer.

2. Left-click on the line to highlight the desired entry.

3. Right-click on the highlighted line to open the context menu.

4. Go to **Copy Cell** or alternatively use the **Ctrl + C** key combination.

5. Add a copy of the entry at the desired location.

   Alternatively, use the **Ctrl + V** keyboard shortcut for this.

Note: Copied entries can now be added into cells that are unlocked for editing. This is only applicable for cells in the **Set Value** column.

15.4 Operation in Runtime

When calling up the **variable diagnosis** screen, the variables configured in the screen switching function are shown in the Runtime.

Carry out the following steps in Runtime to remove or add variables from the variable list:

▸ Click on the **Add…**
The variable selection dialog is opened.

▸ Select the desired variables. Multiple selection is possible.
Note: The variables in the selection area can be sorted and filtered.

- ▶ Click on the Add button to apply the selected variables.

- ▶ Confirm your pre-selection with OK.

- ▶ The variable selection dialog is closed.
  The selected variables are listed in the variable list.

HIDE VARIABLES

- ▶ Select the desired variable in the variable list with a simple right click.

- ▶ Select additional, optional variables with the **CTRL key and a right mouse click** (multi-select).

- ▶ Click on the **Release** button.
  The selected variables are removed from the variable list.

VARIABLE SELECTION DIALOG

The variable selection dialog is called up by clicking on the **Add** button.
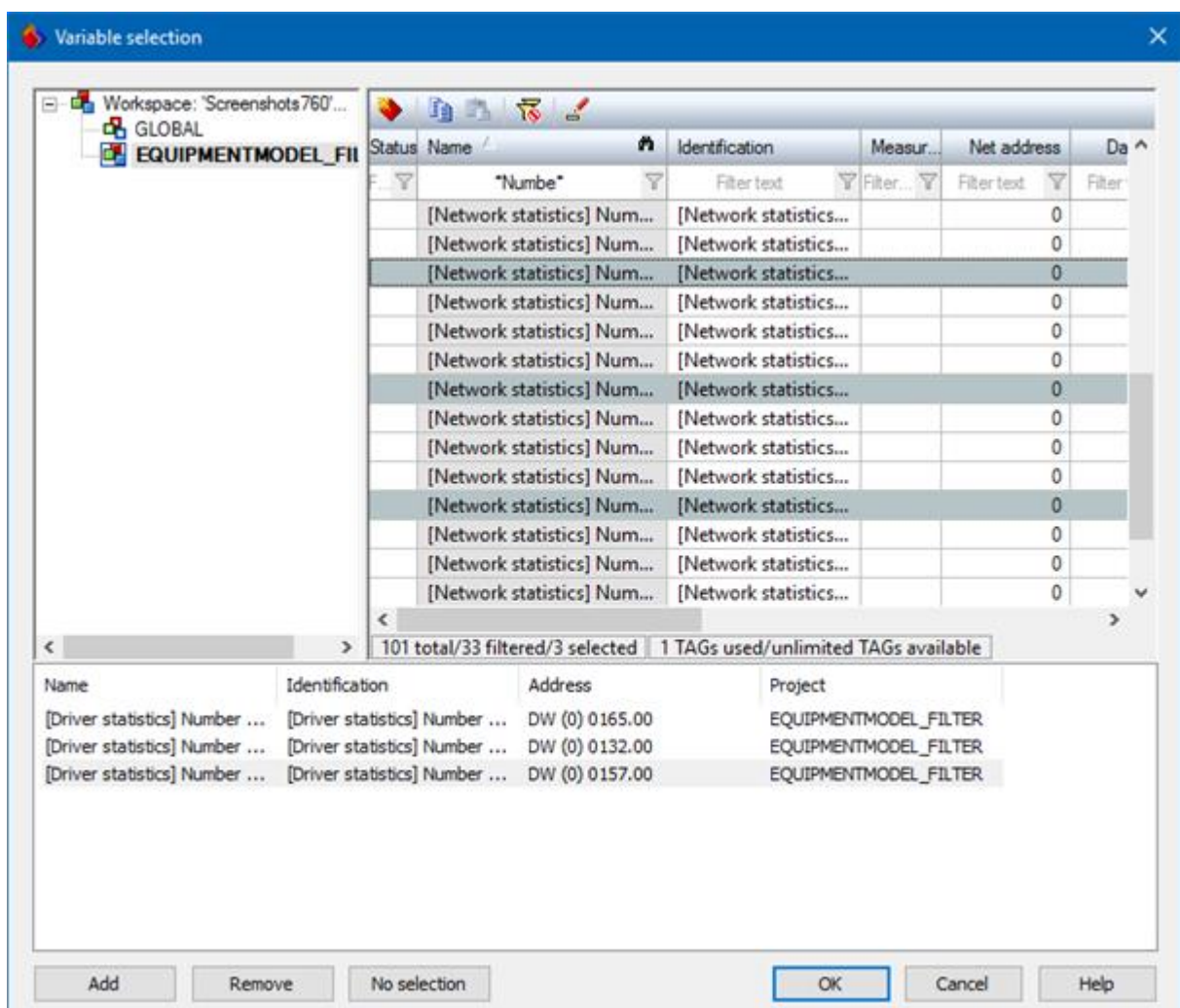
Variables that are already shown in the list are included in the list of variables to be displayed.

Parameter	Description
Workspace	Tree view of the current workspace.
	The current project and integration projects therein are shown.
	Selection of the project by clicking.
Variable list	List of all configured variables of a project.
	This list can be sorted and filtered.
	Additional columns can be displayed by means of a context menu. You get this context menu if you right in a column heading.
	Information about available and selected variables is shown in the footer of the **variable list** as text.
List of variables to be displayed	List of all selected variables.
Add	Adds selected variables from the **variable list** to the **list of variables to be displayed**.
	Note: Variables from the variable list can also be accepted by double clicking.
Remove	Removes selected variables from the **list of variables to be displayed**.
No selection	removes all variables from the **list of variables to be displayed** and closes the dialog.
	No variables are displayed in Runtime. The list is empty.
Up arrow key	The current selection switches to the entry above.
Down arrow key	The current selection switches to the entry below.
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

§§ Trenner

You can copy individual cell content from the variable list in the Runtime:

1. Go to the desired entry with the mouse pointer.

2. Left-click on the line to highlight the desired entry.

3. Right-click on the highlighted line to open the context menu.

4. Go to **Copy Cell** or alternatively use the **Ctrl + C** key combination.

5. Add a copy of the entry at the desired location.

   Alternatively, use the **Ctrl + V** keyboard shortcut for this.

Note: Copied entries can now be added into cells that are unlocked for editing. This is only applicable for cells in the **Set Value** column.

16 External settings

The **External settings** properties group provides properties in order to be able to also use zenon variables in zenon Logic and Service Grid.

▸ **Integrated 61131-3 settings**: Properties in order to also be able to evaluate zenon variables in zenon Logic. Only affects variables that are not based on the **straton** driver.
For details, see the **zenon Logic Workbench basics** manual in the **Use of zenon Variables in zenon Logic** chapter.

▸ **Service Grid settings**: Properties for the use of <CD_PRODUCTNAMME> variables in the Service Grid. These are only available for simple variables.
For details, read the **Service Grid** (on page 221) chapter and the **Service Grid** manual.

16.1 Service Grid

The handling of zenon variables in the Service Grid is configured in this subgroup. It is only available for simple variables. You can read details on the Service Grid in the **Service Grid** manual.

Configurable properties:

▸ **Access permission**: Configuration of which access authorizations there are for a variable in the Service Grid.
Select from drop-down list:

   ▸ *None*: This variable is not available in the Service Grid.

   ▸ *Read*: There is read access to this variable in the Service Grid.

   ▸ *Read and write*: This variable can be read and written in the Service Grid.

   Default: *none*

17 Measuring unit switch

measuring 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 measuring 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.

Note: In order to be able to use measuring unit conversion, at least one base unit and a conversion unit must be set up. Define these units in the project tree under **Variables** and **Measuring Units**.

A base unit can be selected when setting parameters for a variable. It is possible to switch between the different units in the Runtime using the Measurement unit switch (on page 226) function.

PROJECT MANAGER CONTEXT MENU

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

💡 **Information**

Measurement unit switching must not be used together with the variablerw function of the Report Generator. This Report function provides a unit conversion for older projects. If the unit defined in the measuring unit conversion of a variable is changed by the report function in the Runtime, you must carry out the configuration in the measuring unit conversion again.

17.1 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 all as XML	Exports all entries as an XML file.

Menu item	Action
Import XML	Imports measuring units from an XML file.
Editor profile	Opens the drop-down list with predefined editor profiles.
Help	Opens online help.

CONTEXT MENU AND TOOLBAR 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**	Drop-down list with link back to the element from which you can reach the measuring 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 measuring units that are already linked to variables or used in functions must then have the respective variable or function amended manually. See also: Allocate a base unit to a variable (on page 225) Measuring Unit conversion function (on page 226).
Export all as XML	Exports all entries as an XML file.
Import XML	Imports measuring 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.

17.2 Engineer measuring 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 225) as a measuring unit.

To create a new base unit:

Select the **Measuring Units** node in the project manager under **Variables**.

1.   Right-click on **measuring units**.

2.   In the context menu, select the **New Base Unit** command.
     A new entry is created in the list.

3.   Give the measuring unit a name.
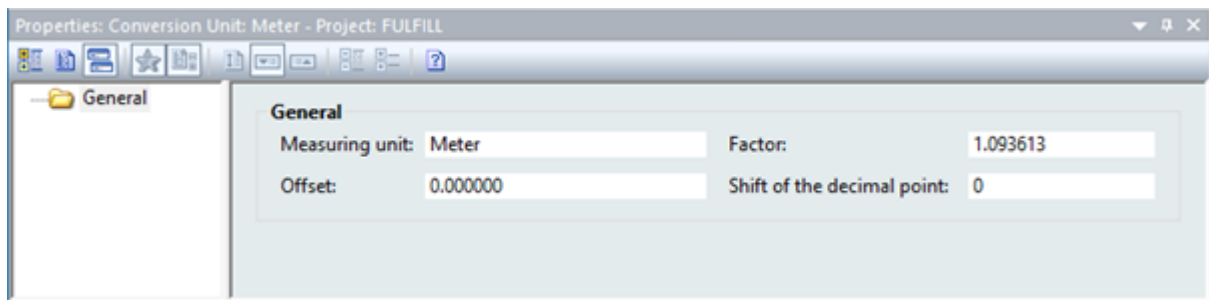
4.   Create the conversion unit.
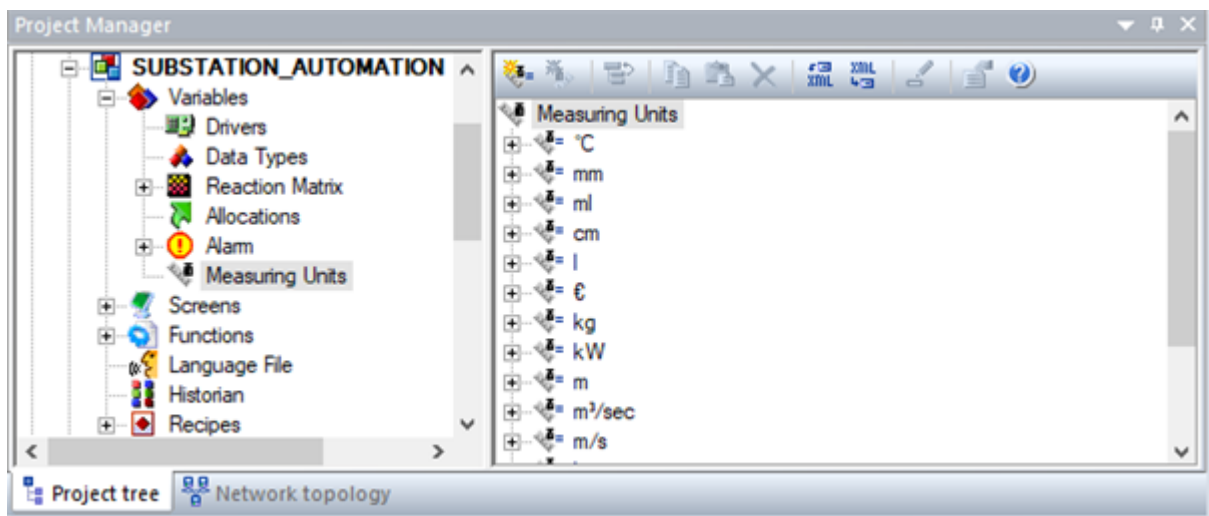
TO CREATE A NEW CONVERSION UNIT

Engineering:

1.   Right-click on a base unit

2.   Select the **New conversion unit** command in the context menu.
     A new entry is created in the list

3.   Give the measuring unit a name.

4.   Define a **Factor** for the conversion.

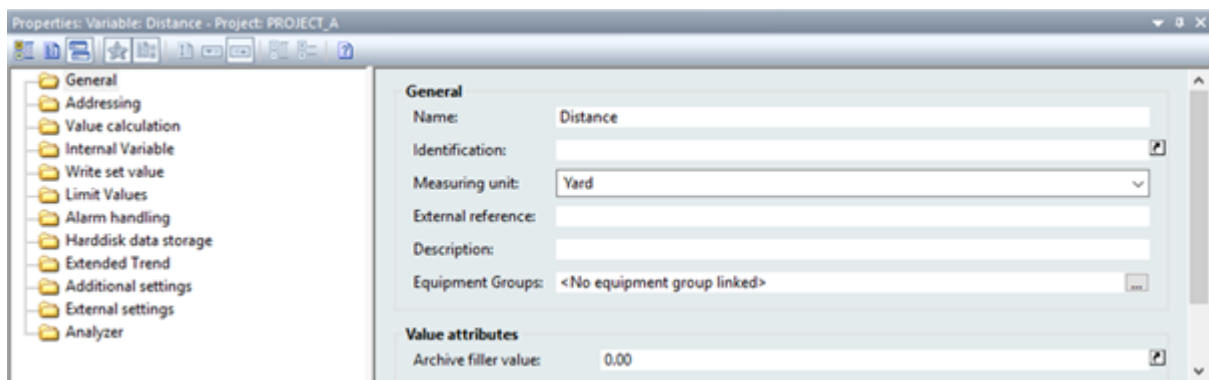5.   Define a value for the **Shift of the decimal point**.

6. Define a value for the **Offset**.

17.3 Allocate a base unit to a variable

Base units are allocated to a variable in the **Measuring unit** property (**General** node).

You are free to name units as you wish here. If the measuring unit conversion is used in the 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 224).

You must create a Unit conversion function (on page 226) in order to be able to convert in the Runtime.

> **⚠Attention**
>
> If a measuring unit is subsequently renamed, variables already linked to this are not automatically renamed.
>
> To rename measuring units already linked:
>
> ▸ select detail view in Project Manager
>
> ▸ select the measuring 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

17.4 Function Measurement unit switch

In order to carry out measuring unit conversion in the Runtime, create a **Measuring unit conversion**:

▶   select the **Functions** node in Project Manager

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

▶   go to group **Variable**

▶   Select the **Measuring unit conversion** function

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

▶   The dialog for the definition of the **Measuring unit conversion** opens.
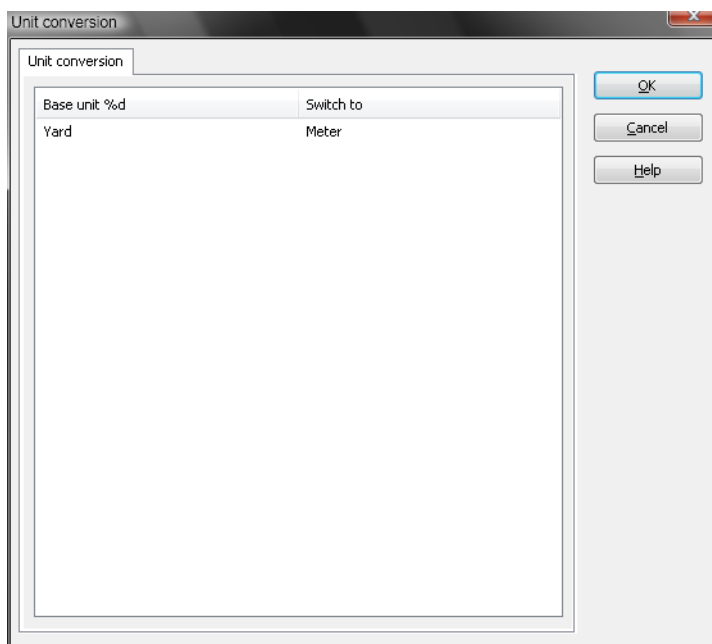
Property	Description
Measuring unit conversion	Dialog for the allocation of conversion units to basic units.
Base Unit	List of the created basic units.
Switch to	Drop-down list for the selection of the conversion unit. The following options are available: ▶ *<No switching>*

Property	Description
	▶ *<Base unit>* ▶ *engineered conversion unit:* The selected conversion unit is output in Runtime. Default: *<No switching>*

⦿ Information

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

⚠Attention

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

17.5 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 226). 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 measuring 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**