zenon
by COPA-DATA

zenon driver manual
MATHDR32

v.8.20

Contents

1 Welcome to COPA-DATA help 5

2 MATHDR32 5

3 MATHDR32 - data sheet 6

4 Driver history 7

5 Configuration 8

5.1 Creating a driver 9

6 Creating variables 12

6.1 Creating variables in the Editor 12

6.2 Addressing 16

6.3 Driver objects and datatypes 17

6.3.1 Driver objects 17

6.3.2 Mapping of the data types 18

6.4 Creating variables by importing 18

6.4.1 XML import 19

6.4.2 DBF Import/Export 20

6.5 Communication details (Driver variables) 25

7 Driver-specific functions 31

7.1 Formulas 32

7.1.1 Assign variable 35

7.1.2 Link types 37

7.1.3 Bit formulas 40

7.1.4 Float formula 41

7.2 Event-triggered calculation of Mathematics variables 64

7.2.1 Activation and deactivation of the calculation 66

7.2.2 Calculation types with special behavior 66

7.2.3 Error messages 67

7.3 Importing Mathematics variables 68

8 Driver command function 68

9 Error analysis 73

9.1 Analysis tool 73

9.2 Driver monitoring 74
9.3 Log server 75
9.4 Check list 75

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 MATHDR32

The 8.20 mathematics driver is used for defining formulas and for calculating them by using data from other process drivers.

> **Information**
>
> The mathematics driver does not use any I/Os.

If a counter variable of the mathematics driver is saved in an archive where saves are carried out in the event of a value change (spontaneous), this can lead to the saving of very large amounts of data.

Workaround: Assign the mathematics variable a simulation variable and save the simulation variable in the archive.
Attention: The simulation variable must be of the *INT* data type (not *REAL*).

Information

The value of the mathematic driver is saved when Runtime is ended normally. To ensure that values are saved if limit values or rema conditions where the counter runs become inactive:

Carry out the **Save AML and CEL ring buffer** or the **Save remanent data** function. In addition, you can also still save the values cyclically (every hour, for example).

3 MATHDR32 - data sheet

General:	
Driver file name	MATHDR32.exe
Driver name	Driver for math. calculations
PLC types	--
PLC manufacturer	zenon system driver; Internal driver

Driver supports:	
Protocol	unknown
Addressing: Address-based	Name based
Addressing: Name-based	--
Spontaneous communication	--
Polling communication	X
Online browsing	--
Offline browsing	--
Real-time capable	--

Driver supports:	
Blockwrite	--
Modem capable	--
RDA numerical	--
RDA String	--
Hysteresis	--
extended API	--
Supports status bit **WR-SUC**	--
alternative IP address	--

Requirements:	
Hardware PC	--
Software PC	--
Hardware PLC	--
Software PLC	--
Requires v-dll	--

Platforms:	
Operating systems	Windows 10; Windows 7; Windows 8; Windows 8.1; Windows Server 2008 R2; Windows Server 2012; Windows Server 2012 R2; Windows Server 2016

4 Driver history

Date	Driver version	Change
07.07.08	400	Created driver documentation

DRIVER VERSIONING

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

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

Example

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

5 Configuration

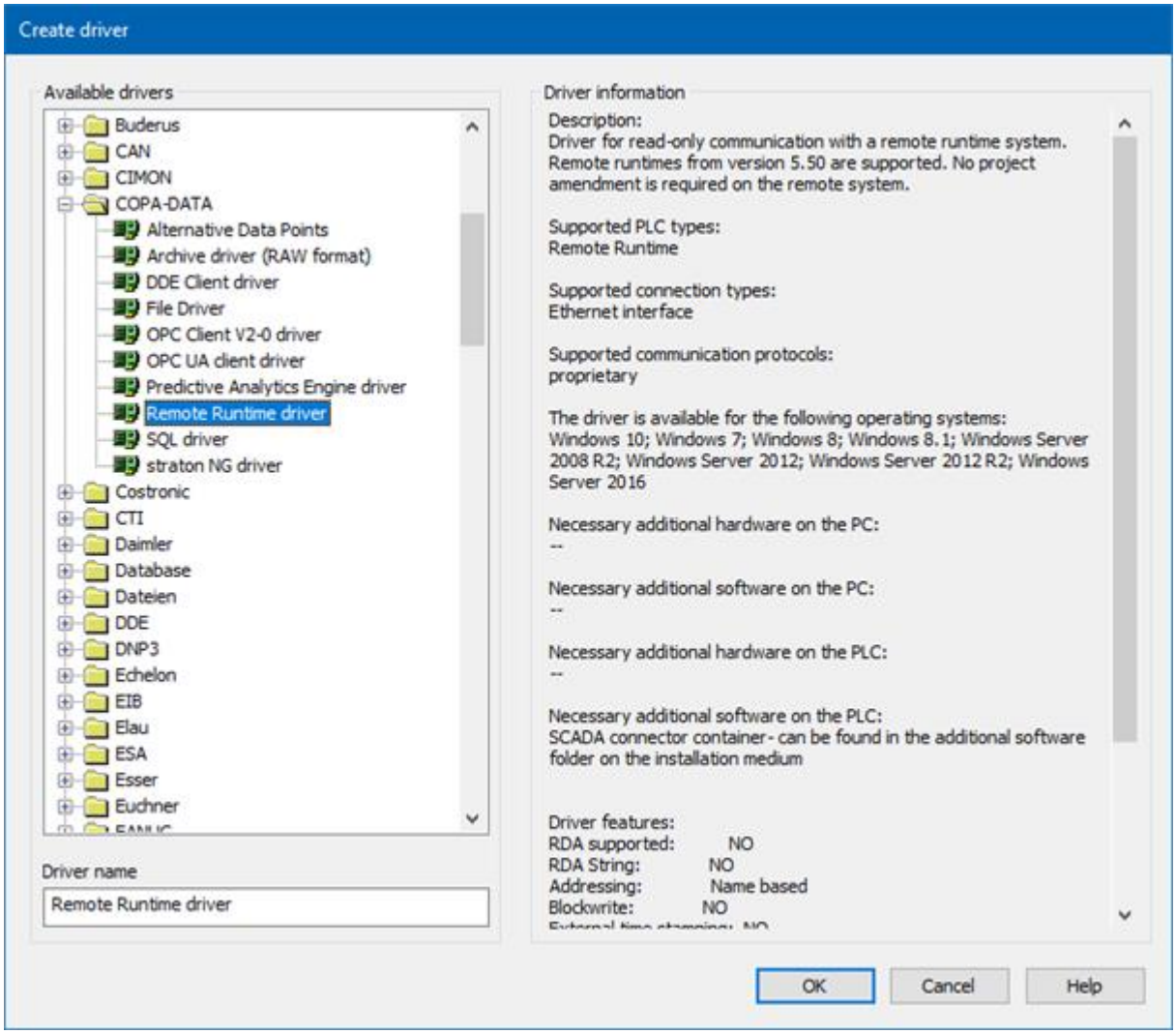
In this chapter you will learn how to use the driver in a project and which settings you can change.

Information

Find out more about further settings for zenon variables in the chapter Variables of the online manual.

5.1 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

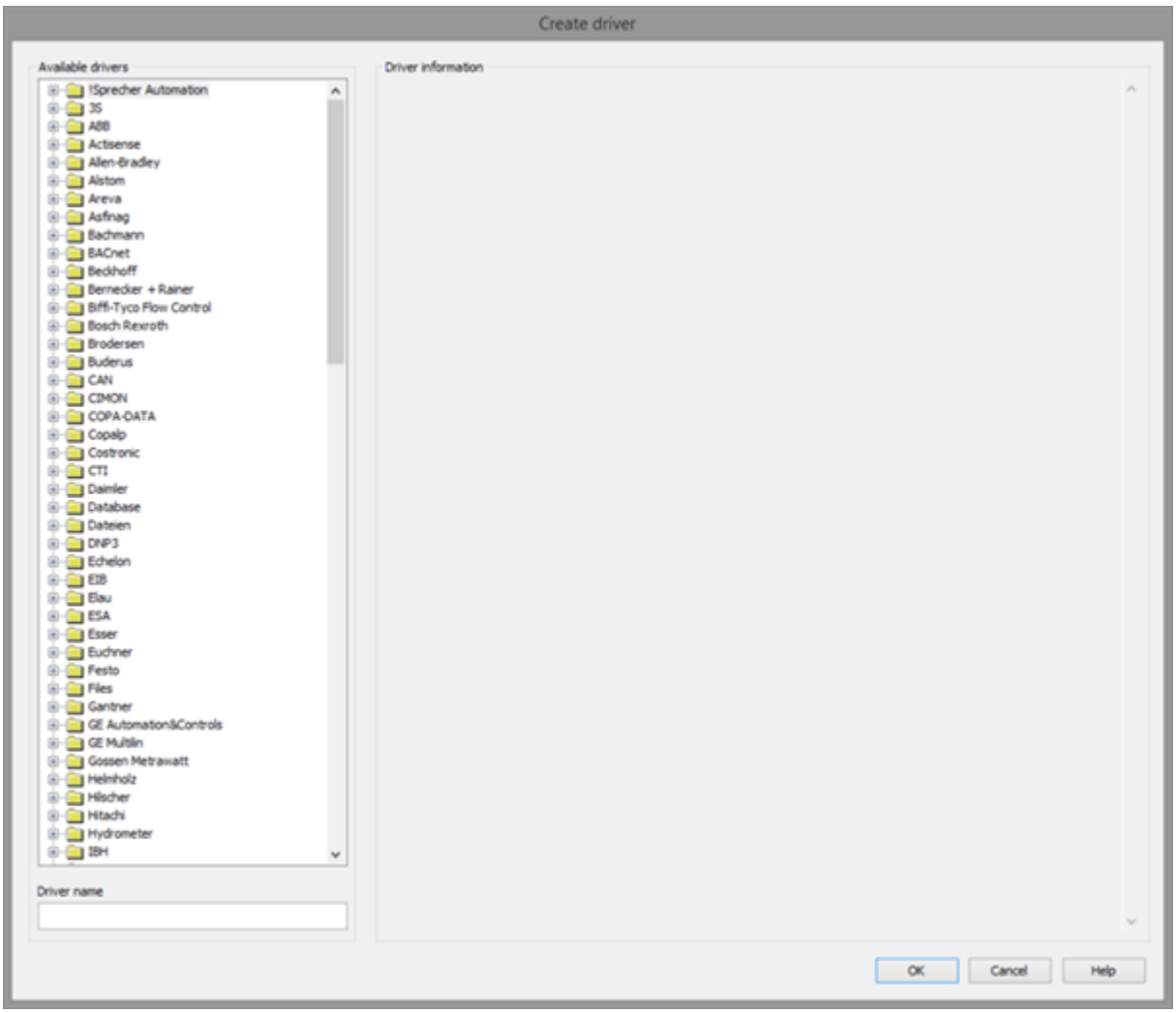
In order to create a new driver:

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**:

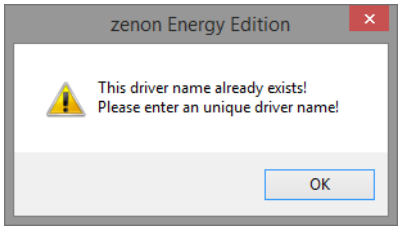
   - The **Driver name** must be unique.
   If a driver is used more than once in a project, a new name has to be given each time. This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
   - The **Driver name** is part of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

6 Creating variables

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

6.1 Creating variables in the Editor

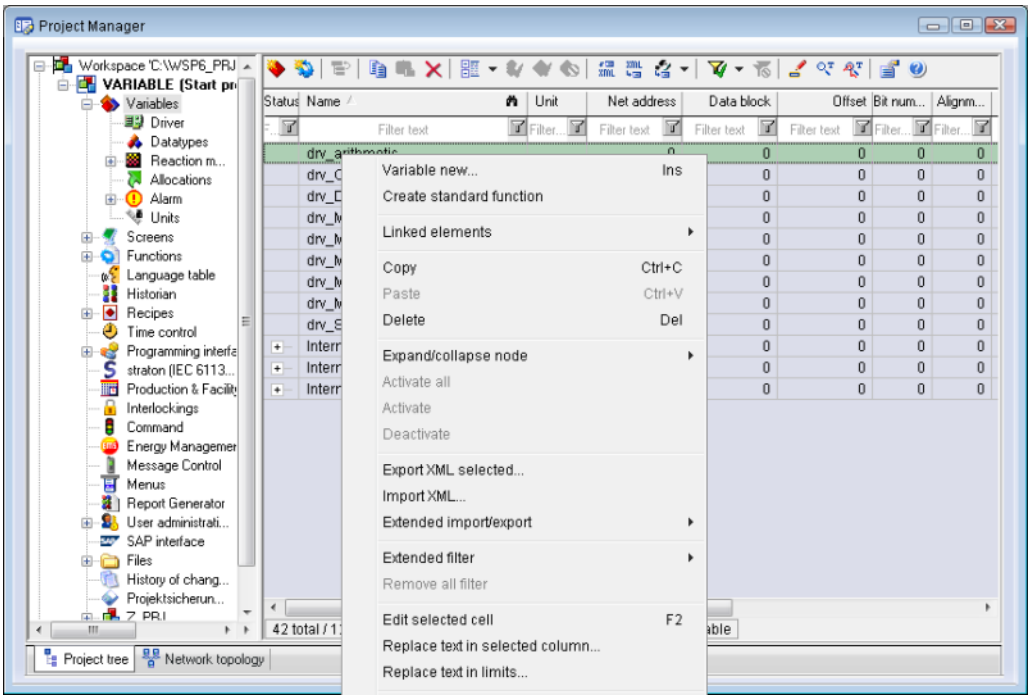
Variables can be created:

- as simple variables
- in arrays
- as structure variables

VARIABLE DIALOG

To create a new variable, regardless of which type:

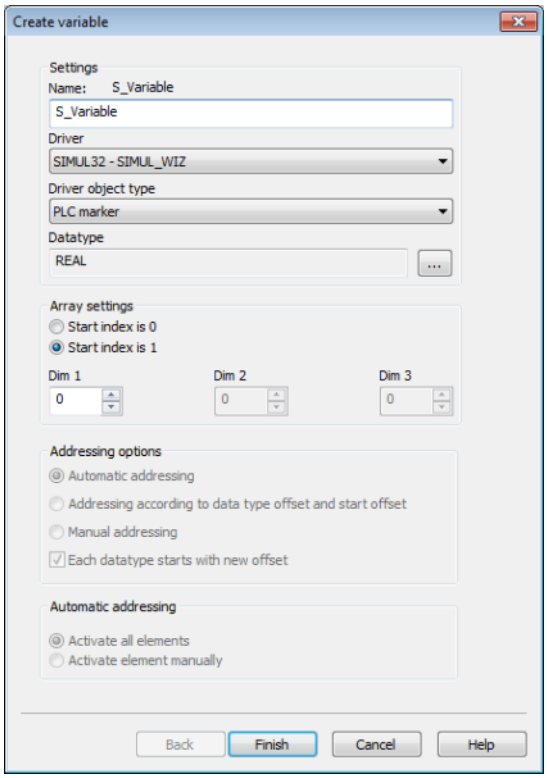
1. Select the **New variable** command in the **Variables** node in the context menu



   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	Select the desired data type. Click on the ... button to open the selection dialog.
Array settings	Expanded settings for array variables. You can find details in the

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

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.

6.2 Addressing

Group/Property	Description
General	Property group for general settings.
Name	Freely definable name. **Attention:** For every zenon project the name must be unambiguous.
Identification	Freely definable identification. E.g. for Resources label, comments, ...
Addressing	
Net address	not used for this driver
Data block	not used for this driver
Offset	not used for this driver
Alignment	not used for this driver
Bit number	not used for this driver
String length	not used for this driver
Driver connection/Data Type	Data type of the variable. Is selected during the creation of the variable; the type can be changed here. **Attention:** If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.
Driver connection/Driver Object Type	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
Driver connection/Priority	not used for this driver The driver does not support cyclically-poling communication in priority classes.

MATHEMATICS VARIABLES

The mathematics driver is used for defining formulas (linkage type of arithmetics, trigonometry, etc.) and calculating them by using the data from other process drivers. From the point of view of zenon,

the Mathematics driver is a "normal" process driver. Variables are defined in the Editor (mathematics variable). The calculated values of the Mathematics variables are provided in the online operation. In order to define and use Mathematics variables, the Mathematics driver must be loaded first.

A formula must be defined for each mathematical variable of the Mathematics driver. Each formula must have at least one source variable. Any previously defined variable (even previously defined Mathematics variables) can be used as a source variable. The link is made with the signal resolution (technical values) of the variables.

Attention

You can configure a maximum of 4096 Mathematics variables with formulas. If you use more formulas, only the first 4096 will be executed in the Runtime. Any further formulas will be ignored.

You can sum up a maximum of 4096 numerical constants among all Mathematics formulas. Any further constants will be assumed as 0 during Runtime.

If you require a larger number of variables/constants, we recommend the use of zenon Logic.

Hint: zenon Logic is already included in zenon. In case zenon Logic is not included in your license, contact the distributor that is responsible for you.

Information

Up to 99 variables can be linked in one formula. X01 to X99. The length of the formula must not exceed 4096 characters.

6.3 Driver objects and datatypes

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

6.3.1 Driver objects

The following object types are available in this driver:

DRIVER OBJECT TYPES AND SUPPORTED IEC DATA TYPES FOR PROCESS VARIABLES IN ZENON

Driver Object Type	Channel type	Read	Write	Supported data types	Comment
Formula	17	X	--	*BOOL, LREAL*	

CHANNEL TYPE

The term **Kanaltyp** is the internal numerical name of the driver object type. It is also used for the extended DBF import/export of the variables.

"**Kanaltyp**" is used for advanced CSV import/export of variables in the "**HWObjectType**" column.

6.3.2 Mapping of the data types

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

MAPPING OF THE DATA TYPES FROM THE PLC TO ZENON DATA TYPES

PLC	zenon	Data type
BOOL	BOOL	8
LREAL	LREAL	5

DATA TYPE

The term **data type** is the internal numerical identification of the data type. It is also used for the extended DBF import/export of the variables.

6.4 Creating variables by importing

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

Information

You can find details on the import and export of variables in the Import-Export manual in the Variables section.

6.4.1 XML import

During XML import of variables or data types, these are first assigned to a driver and then analyzed. Before import, the user decides whether and how the respective element (variable or data type) is to be imported:

- *Import*:
  The element is imported as a new element.
- *Overwrite*:
  The element is imported and overwrites a pre-existing element.
- *Do not import*:
  The element is not imported.

Note: The actions and their durations are shown in a progress bar during import. The import of variables is described in the following documentation. Data types are imported along the same lines.

REQUIREMENTS

The following conditions are applicable during import:

- **Backward compatibility**

  At the XML import/export there is no backward compatibility. Data from older zenon versions can be taken over. The handover of data from newer to older versions is not supported.

- **Consistency**

  The XML file to be imported has to be consistent. There is no plausibility check on importing the file. If there are errors in the import file, this can lead to undesirable effects in the project.

  Particular attention must be paid to this, primarily if not all properties exist in the XML file and these are then filled with default values. E.g.: A binary variable has a limit value of *300*.

- **Structure data types**

  Structure data types must have the same number of structure elements.
  Example: A structure data type in the project has 3 structure elements. A data type with the same name in the XML file has 4 structure elements. Then none of the variables based on this data type in the file are imported into the project.

Hint

You can find further information on XML import in the **Import - Export** manual, in the **XML import** chapter.

6.4.2 DBF Import/Export

Data can be exported to and imported from dBase.

Information

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

IMPORT DBF FILE

To start the import:

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

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

Information

Note:

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

EXPORT DBF FILE

To start the export:

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

Attention

DBF files:

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

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

Information

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

FILE STRUCTURE OF THE DBASE EXPORT FILE

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

Attention

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

DBF files must:

- conform with their name to the 8.3 DOS format (8 alphanumeric characters for name, 3 characters for extension, no space)
- Be stored close to the root directory (Root)

STRUCTURE

Identification	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered

Identification	Type	Field size	Comment
			manually). The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
KANAL_D	Log	1	The variable is deleted with the *1* entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the **MAX_LAENGE** entry in the **project.ini** file.
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Net address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager
LES_SCHR	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTTYP and DATENTYP

Identification	Type	Field size	Comment
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix
ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the Standby Server in redundant networks
RESOURCE	C	128	Resources label. Free string for export and display in lists.

Identification	Type	Field size	Comment
			The length can be limited using the MAX_LAENGE entry in **project.ini**.
ADJWVBA	R	1	Non-linear value adaption: *0*: Non-linear value adaption is used *1*: Non-linear value adaption is not used
ADJZENON	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
ZWREMA	N	16	Linked counter REMA.
MAXGRAD	N	16	Gradient overflow for counter REMA.

Attention

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

LIMIT VALUE DEFINITION

Limit definition for limit values *1* to *4*, or status *1* to *4*:

Identification	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	technical value or ID number of a linked variable for a dynamic limit value (see VARIABLEx) (if VARIABLEx is *1* and here it is *-1*, the existing variable linkage is not overwritten)
SCHWWERT1	F	16	Threshold value for limit value
HYSTERESE1	F	14	Is not used
BLINKEN1	R	1	Set blink attribute
BTB1	R	1	Logging in CEL
ALARM1	R	1	Alarm

Identification	Type	Field size	Comment
DRUCKEN1	R	1	Printer output (for CEL or Alarm)
QUITTIER1	R	1	Must be acknowledged
LOESCHE1	R	1	Must be deleted
VARIABLE1	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
FUNC1	R	1	Functions linking
ASK_FUNC1	R	1	Execution via Alarm Message List
FUNC_NR1	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
A_GRUPPE1	N	10	Alarm/Event Group
A_KLASSE1	N	10	Alarm/Event Class
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Color as Windows coding
GRENZTXT1	C	66	Limit value text
A_DELAY1	N	10	Time delay
INVISIBLE1	R	1	Invisible

Expressions in the column "Comment" refer to the expressions used in the dialog boxes for the definition of variables. For more information, see chapter Variable definition.

6.5 Communication details (Driver variables)

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

Name from import	Type	Offset	Description
LineStateRingBack	*BOOL*	24.5	While establishing the connection
LineStateBusy	*BOOL*	24.6	Target station is busy
LineStateSpecialInfo	*BOOL*	24.7	Special status information received
LineStateConnected	*BOOL*	24.8	Connection established
LineStateProceeding	*BOOL*	24.9	Dialing completed
LineStateOnHold	*BOOL*	24.10	Connection in hold
LineStateConferenced	*BOOL*	24.11	Connection in conference mode.
LineStateOnHoldPendConf	*BOOL*	24.12	Connection in hold for conference
LineStateOnHoldPendTransfer	*BOOL*	24.13	Connection in hold for transfer
LineStateDisconnected	*BOOL*	24.14	Connection terminated.
LineStateUnknow	*BOOL*	24.15	Connection status unknown
ModemStatus	*UDINT*	24	Current modem status
TreiberStop	*BOOL*	28	Driver stopped For *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

Name from import	Type	Offset	Description
			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. 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

Name from import	Type	Offset	Description
ComPort	*UINT*	9	Number of the serial interface.
Baudrate	*UDINT*	10	Baud rate of the serial interface.
Parity	*SINT*	11	Parity of the serial interface
ByteSize	*USINT*	14	Number of bits per character of the serial interface Value = *0* if the driver cannot establish any serial connection.
StopBit	*USINT*	13	Number of stop bits of the serial interface.
Autoconnect	*BOOL*	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	*STRING*	17	Current telephone number
ModemHwAdr	*DINT*	21	Hardware address of current telephone number
RxIdleTime	*UINT*	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)
WriteTimeout	*UDINT*	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	*UDINT*	20	Number of ringing tones before a call is accepted
ReCallIdleTime	*UINT*	53	Waiting time between calls in seconds (s).
ConnectTimeout	*UINT*	54	Time in seconds (s) to establish a connection.

STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	*UDINT*	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	*UDINT*	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	*UDINT*	40	Longest time in milliseconds (ms) that is required to read a data block.

Name from import	Type	Offset	Description
MinBlkReadTime	*UDINT*	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	*UDINT*	33	Number of writing errors
ReadSucceedCount	*UDINT*	35	Number of successful reading attempts
MaxCycleTime	*UDINT*	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	*UDINT*	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	*UDINT*	26	Number of writing attempts
ReadErrorCount	*UDINT*	34	Number of reading errors
MaxUpdateTimeNormal	*UDINT*	56	Time since the last update of the priority group **Normal** in milliseconds (ms).
MaxUpdateTimeHigher	*UDINT*	57	Time since the last update of the priority group **Higher** in milliseconds (ms).
MaxUpdateTimeHigh	*UDINT*	58	Time since the last update of the priority group **High** in milliseconds (ms).
MaxUpdateTimeHighest	*UDINT*	59	Time since the last update of the priority group **Highest** in milliseconds (ms).
PokeFinish	*BOOL*	55	Goes to *1* for a query, if all current pokes were executed

ERROR MESSAGE

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

Name from import	Type	Offset	Description
RdErrBlockCount	*UINT*	44	Number of blocks to read when the last reading error occurred.
RdErrHwAdresse	*DINT*	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	*UDINT*	46	Block number when the last reading error occurred.
RdErrMarkerNo	*UDINT*	47	Marker number when the last reading error occurred.
RdErrSize	*UDINT*	48	Block size when the last reading error occurred.
DrvError	*USINT*	25	Error message as number
DrvErrorMsg	*STRING*	30	Error message as text
ErrorFile	*STRING*	15	Name of error log file

7 Driver-specific functions

The configuration of the mathematics driver is predefined and cannot be changed. The **update time** of the driver is *500 ms*.

Attention

If a counter variable - with the formula: *Count()* - of the mathematics driver is saved in an archive for which *saving is carried out in the event of a value change* (spontaneous), this can lead to very large amounts of data being saved.

Workaround: Assign the mathematics variable of a simulation variable to the (**SIMUL32** driver, *HD* **driver object type**) and save the simulation variable in the archive.
Attention: The simulation variable must be an integer data type (e.g. *INT*, *DINT*), and not *REAL*.

MATHEMATICS DRIVER IN THE ZENON NETWORK

The driver carries out the calculations on the primary server. These results are distributed to all the clients in the network, including the standby server.

On the standby server, the driver does the calculations in the background, based on the current values of the variables as they are distributed by the primary server; not on the basis of the values of the variables in the internal buffer of the standby server. Only if the primary server fails does the standby server use the values from its process drivers for the calculations.

If the local (not process) variables are linked in formulas, the result of the standby server can be different than that of the primary server, which however cannot be seen as long as it is the standby server. After redundancy switching, the standby server that is promoted to the role of primary server can then switch the mathematic variables to other values - based on its own calculations.

Note: the variables of the internal driver are set to the **Calculation** = *local* property by default, and even some of the SYSDRV variables are local.

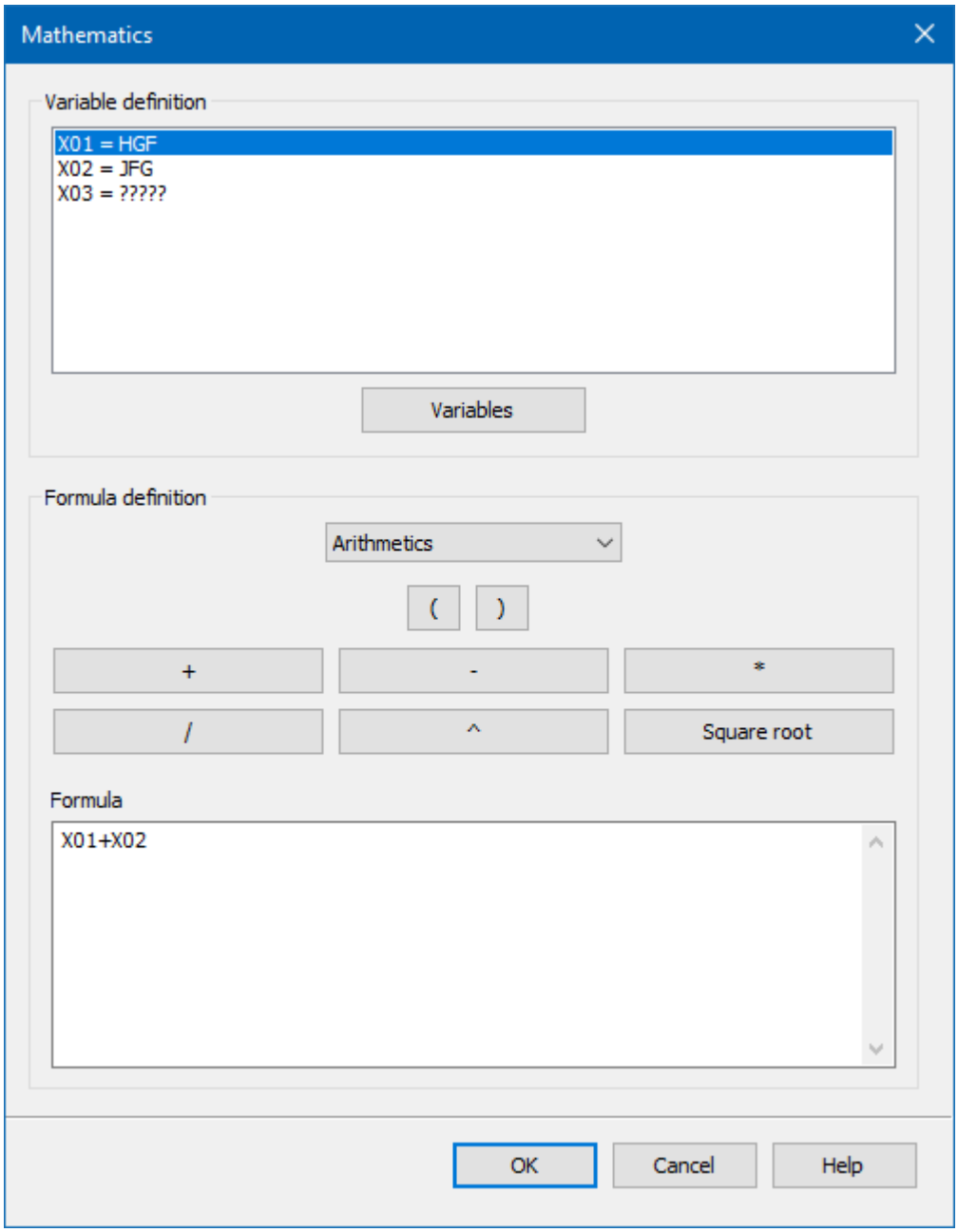
The driver supports the following functions:

7.1 Formulas

To enter formulas

- there must be a variable of the type MATHDR32 Mathematics driver
- open the node **Value calculation** in the properties of the variable
- Click into the empty field next to the **Formula** property.

- the dialog for entering formulas opens



Formulas can be entered as Float formulas (on page 41) or as Bit formulas (on page 40):

- Float formulas:
  - Direct input in the section **Formula**

    or

  - Select a variable in the window **Variable definition** and select a formula type by clicking on the corresponding buttons for functions in the section **Formula definition**.

- Bit formulas:
  Direct input in the **Formulas** section after selecting **Boolean algebra**. If the variable was created as a binary variable, you will only be able to use **Boolean algebra**.

  ### Information

  Up to 99 variables can be linked in one formula. X01 to X99. The length of the formula must not exceed 4096 characters.

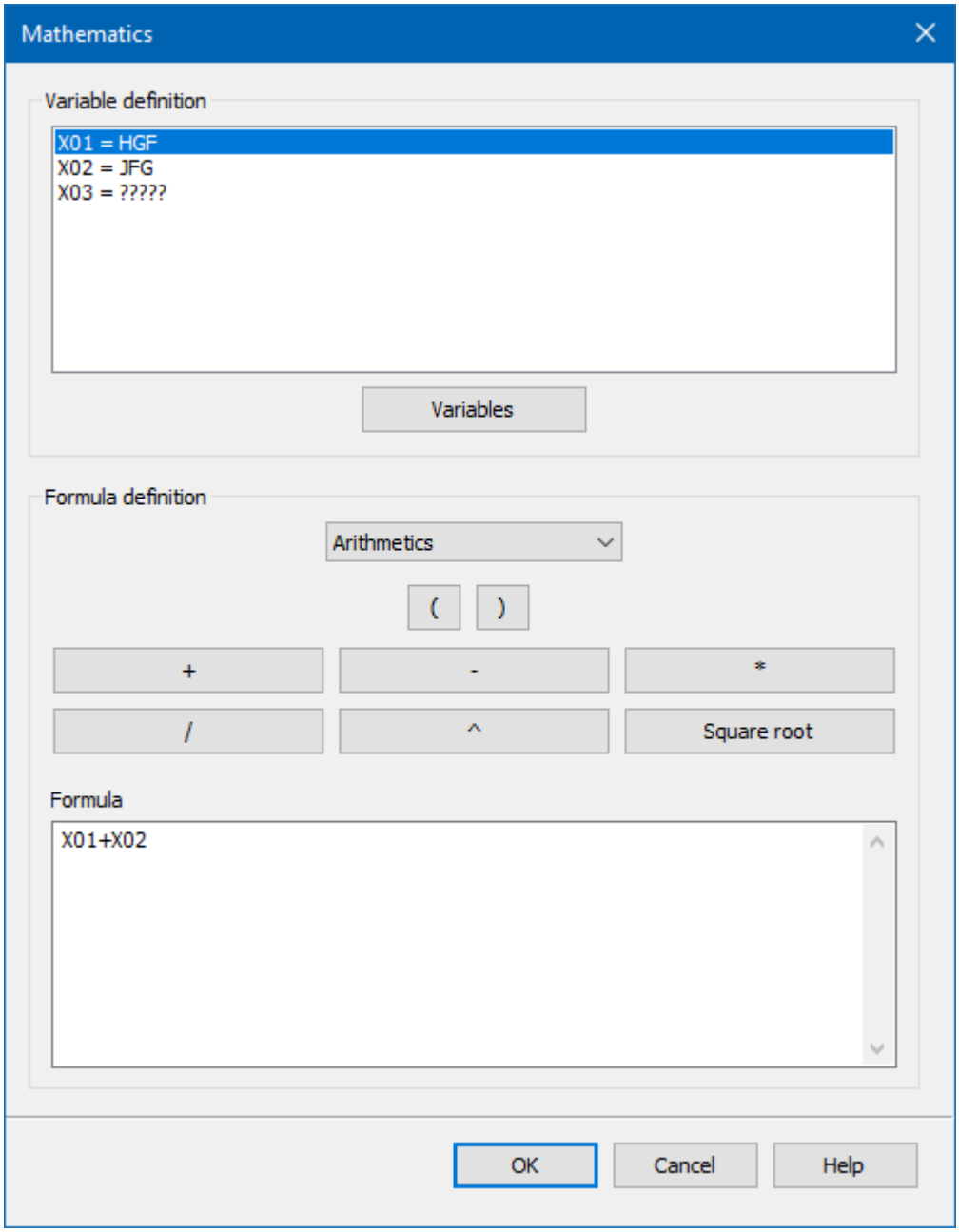
Note: When selecting a function, the necessary parentheses are created automatically and the cursor is placed between the parentheses.

Attention

Avoid recursive calls, such as a *statistic*-type mathematic variable, which counts itself in **X01**. Recursive calls can lead to the whole system becoming unstable.

7.1.1 Assign variable

Source variables for formula input are displayed in the field **Variable definition** .



To add a new variable, change an existing variable or to delete a variable, open the variable selection:

1. Click on the button Variables.

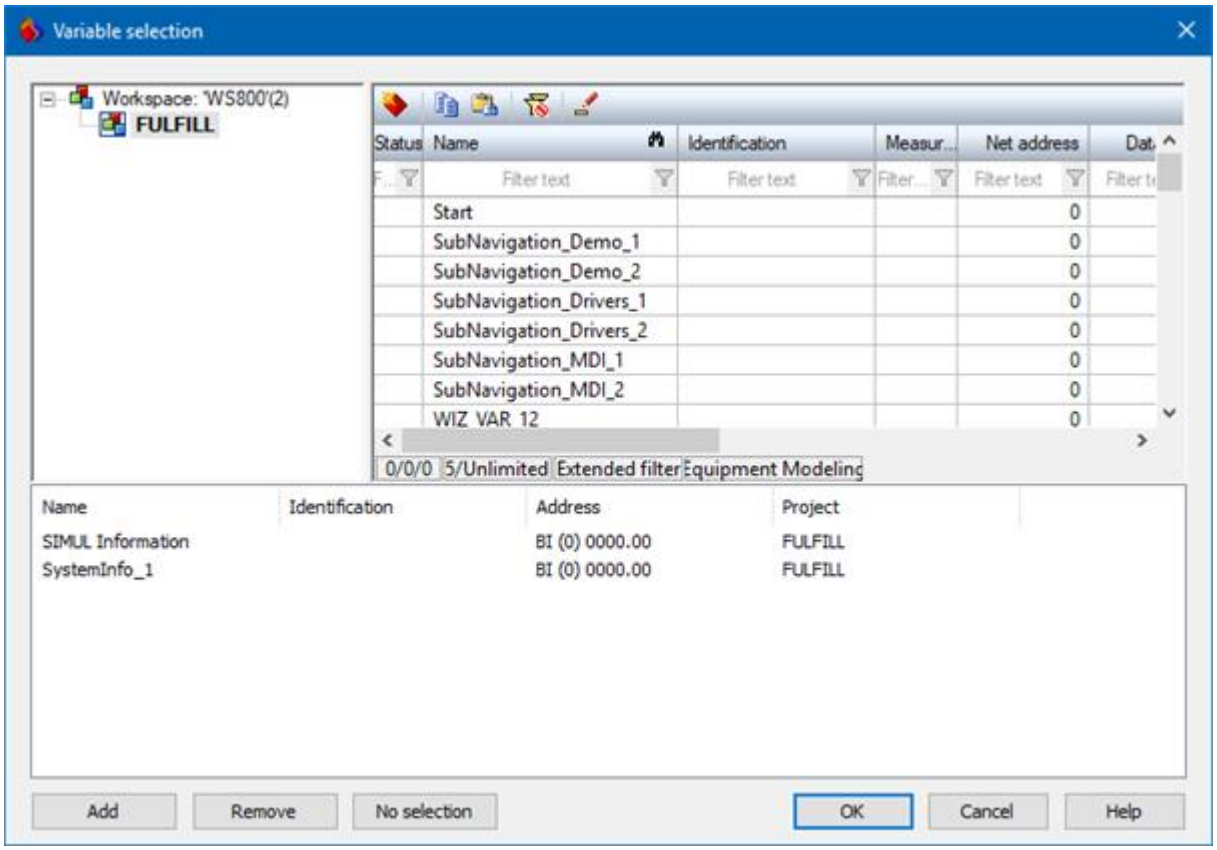
   Or:

   Double-click on the last consecutive number with a question sign as the definition set in the field **Variable definition**

The dialog for selecting variables will be opened.

2. Add new variables or delete existing ones from the list.

ADD VARIABLES



Element	Description
Project tree	Definition of the project from which the variable shall be selected.
Selection window	Selection of the variables: ▸ Double click the selected variable in order to add it to the variable list. ▸ You can move the selected variable to the variable list via Drag&Drop ▸ Select the desired variable. With the help of **Ctrl** or **Shift** multi-selection is possible. By clicking **Add** the selected variables are added to the variable list.
Variable List	Lists all selected variables.
Add	Adds the currently selected variable of the selection window to the variable list.

Element	Description
Remove	Removes the variables which are selected in the variable list from the list.
No selection	Depending on the element: ▸ the dialog is canceled ▸ certain links such as lot variables in archiving can be released

▸

7.1.2 Link types

For entering formulas, you can use different functions from the following mathematical areas:

- **Arithmetics** (combinable with other areas)
- **Trigonometry** (combinable with other areas)
- **Boolean algebra** (combinable with other areas)
- **Statistics** (not combinable)
- **Cross calculation** (not combinable)
- **Data reduction** (not combinable)
- **Comparison** (combinable with other areas)
- **Counted measurand processing** (only SICAM 230)

Range	Expression	Function
Arithmetics	Phases	
	+	Addition
	-	Subtraction
	*	Multiplication
	/	Division
	^	Exponential calculation
	SQRT	Square root
	()	Brackets

Range	Expression	Function
Trigonometry	Trigonometric functions: Value is regarded as radian (0..x*PI)	
	sin	Sine
	cos	cosine
	tan	Tangent
	sinh	hyperbolic sine
	cosh	hyperbolic cosine
	tanh	hyperbolic tangent
	()	Brackets
Boolean algebra	Logical links: Variables will only be checked for the logical value '0'; if the value does not equal '0', it will be considered as '1'. As opposed to Bit formulas, the technical range of the Mathematics variables can be modified by a stretch factor (not equal '0' or '1').	
	AND	logical 'AND'
	OR	logical 'OR'
	XOR	logical 'EXCLUSIVE OR'
	NOT	Negation
	()	Brackets
Statistics	The value is only available at the end of the calculation period (number of events).	
	GltndMW	Moving average for changes
	GltndStdabw	moving deviation of changes
	Counter	counter function
	()	Brackets

Range	Expression	Function
Cross calculation		
	Quer MW	Calculation of the average of several channels at the same time
	()	Brackets
Data reduction		
	TMittelung	Average over time and events
	Sum	Sum over time and events
	Max	Maximum over time and events
	Min	Minimum over time and events
	Dif	Difference over time
	()	Brackets
Comparison		
	<	less than
	>	greater than
	<=	Less than or equal
	>=	greater or equal
	<>	less or greater
	=	equal
	()	Brackets
Counting value processing	For linkages with time or event reference (statistics, data reduction) it is necessary to make additional entries. You can make them when ending the link.	Only for SICAM 230.

7.1.3 Bit formulas

Bit formulas are suitable for formula results with LReal mathematic variables that only have a logical state low or high. In contrast to float formulas with Boolean processing (on page 42), the raw value is already predefined (0,1).

Entering Bit formulas:

- mapping the variable name
- Optional: inputs for technical operation, limit values and update behavior.
- Allocating formulas

Only Boolean operators are available for calculations (see chapter Boolean algebra (on page 42).)

The mathematics driver also allows you to define values (e.g. multi-bit) bitwise by means of Boolean linkings. To do this, the bit pattern of the variables to be linked has to be entered decimally in the bit formula after the operator (for example **Bit0** and **Bit1** ->*3*).

Syntax	Description
(X01)NAND(X02)	NOT AND
(X01)NOR(X02)	NOT OR
(X01&3)	**Bit0**=*1* and **Bit1**=*1* **AND** are linked to the value of the variables X01.
(X01|3)	**Bit0**=*1* and **Bit1**=*1* **OR** are linked to the value of the variables X01.
(X01#3)	**Bit0**=*1* and **Bit1**=*1* **XOR** are linked to the value of the variables X01.

EXAMPLES

Formula	Values (binary)	Result	= numerical	= Bool
for: X01=9 (1001b)				
X01|5	*1001|101*	**1101b**	*13*	*true*
X01&5	*1001&101*	**0001b**	*1*	*true*
X01#5	*1001#101*	**1100b**	*12*	*true*
for X01=4 (0100b)				
X01&2	*0100&010*	**0000b**	*0*	*false*

7.1.4 Float formula

The Mathematics interpreter in zenon provides functions from the following areas for float formulas:

- Arithmetics (on page 41)
- Trigonometry (on page 42)
- Boolean algebra (on page 42)
- Statistics (on page 43)
- Cross calculation (on page 49)
- Data reduction (on page 50)
- Comparison (on page 62)
- Counting value processing (on page 63)

as well as some other float formulas (on page 63) that must be entered directly.

7.1.4.1 Arithmetics

Select **Arithmetics** from the drop-down list in the formula definition section. The following basic functions are available:

Expression	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponential calculation
SQRT	Square root
()	Brackets

Arithmetic functions can be combined with:

- Trigonometry (on page 42)
- Boolean algebra (on page 42)
- Comparison (on page 62)

7.1.4.2 Trigonometry

Select **Trigonometry** from the drop-down list in the **Formula definition** section. The following trigonometric functions are available; the value will be handled as a radian (0...x*PI):

Expression	Function
sin	Sine
cos	cosine
tan	Tangent
sinh	hyperbolic sine
cosh	hyperbolic cosine
tanh	hyperbolic tangent
()	Brackets

Trigonometry functions can be combined with:

- Arithmetics (on page 41)
- Boolean algebra (on page 42)
- Comparison (on page 62)

7.1.4.3 Boolean algebra

Select **Boolean algebra** from the drop-down list in the **Formula definition**section. The following expressions are available:

Expression	Function
AND	logical 'AND'
OR	logical 'OR'
XOR	logical 'EXCLUSIVE OR'
NOT()	Negation **Attention:** The expression after *NOT* must be in brackets.
()	Brackets

For logical calculations, the variables will only be checked for the logical value '0'. If the value is not '0', it will be assumed as '1'. As opposed to Bit formulas (on page 40), the technical range of the Mathematics variables can be modified by a stretch factor (not equal '0' or '1').

Boolean alegbra can be combined with:

- Arithmetics (on page 41)
- Trigonometry (on page 42)
- Comparison (on page 62)

7.1.4.4 Statistics

Select **Statistics** from the drop-down list in the **Formula definition** section. The following basic functions are available:

Expression	Function
GltndMW	Moving average for changes (on page 43)
GltndStdabw	floating standard deviation (on page 45)
Counters	counter function (on page 46)
()	Brackets

The value is only available at the end of the calculation period (number of events).

Statistical functions cannot be combined with any further functions.

Information

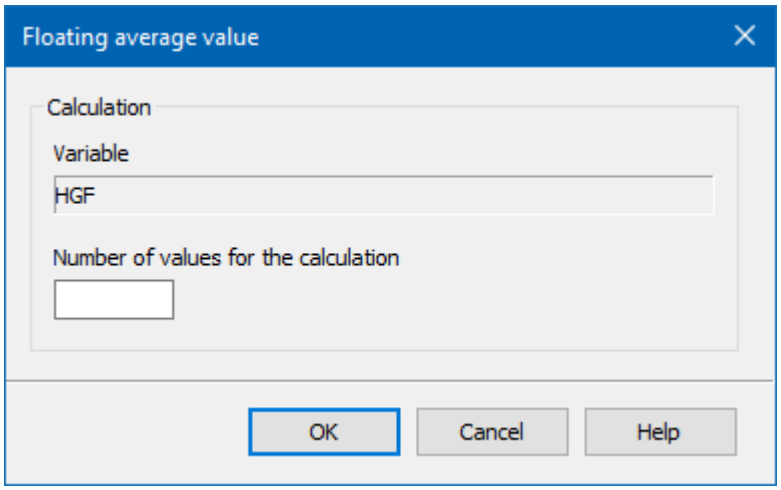
The value of the mathematic driver is saved when Runtime is ended normally. To ensure that values are saved if limit values or rema conditions where the counter runs become inactive:

Carry out the **Save AML and CEL ring buffer** or the **Save remanent data** function. In addition, you can also still save the values cyclically (every hour, for example).

7.1.4.4.1 Floating average value

The standard deviation over a defined number of values is calculated.

A dialog to input the number of values is opened when the function is confirmed:



Option	Description
Calculation	Configuration of the calculation for the given value.
Number of values for calculation	Number of values for which the calculation is carried out. After this number has been reached, the calculation is carried out and moved by one with further value.
OK	Accepts input and closes dialog as well as formula input.
Cancel	Discards input, closes dialog and returns to formula input.
Help	Opens online help.

Example

10 values are set as the requirement.

Procedure in the Runtime:

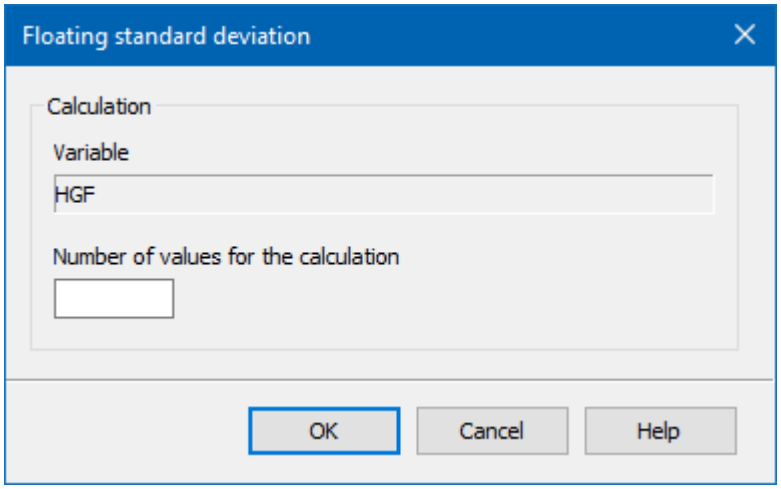
- Starting with the initial value, *10* values are waited for and then the average is calculated from this.
- After the first 10 values, the first value from the calculation is removed and the new 11th value is used for the calculation, and so on. Value included in the calculation etc.

The area for calculation therefore also moves with the growing number of the values.

7.1.4.4.2 Floating standard deviation

The standard deviation over a defined number of values. The first output of the value occurs when all changes are present.

A dialog to input the number of values is opened when the function is confirmed:



Option	Description
Calculation	Configuration of the calculation for the given value.
Number of values for calculation	Number of values for which the calculation is carried out. After this number has been reached, the calculation is carried out and moved by one with further value.
OK	Accepts input and closes dialog as well as formula input.
Cancel	Discards input, closes dialog and returns to formula input.
Help	Opens online help.

Example

10 values are set as the requirement.

Procedure in the Runtime:

- Starting with the initial value, *10* values are waited for and then the standard deviation is calculated from this.
- After the first 10 values, the first value from the calculation is removed and the new 11th value is used for the calculation, and so on. Value included in the calculation etc.

The area for calculation therefore also moves with the growing number of the values.

7.1.4.4.3Counter

The counter function allows you to increase a counter variable every time the Rema status or the limit value of a source variable is reached. With this, the operating hours of equipment can be counted, for example, and the signal for required maintenance work can be given with the counter variables.

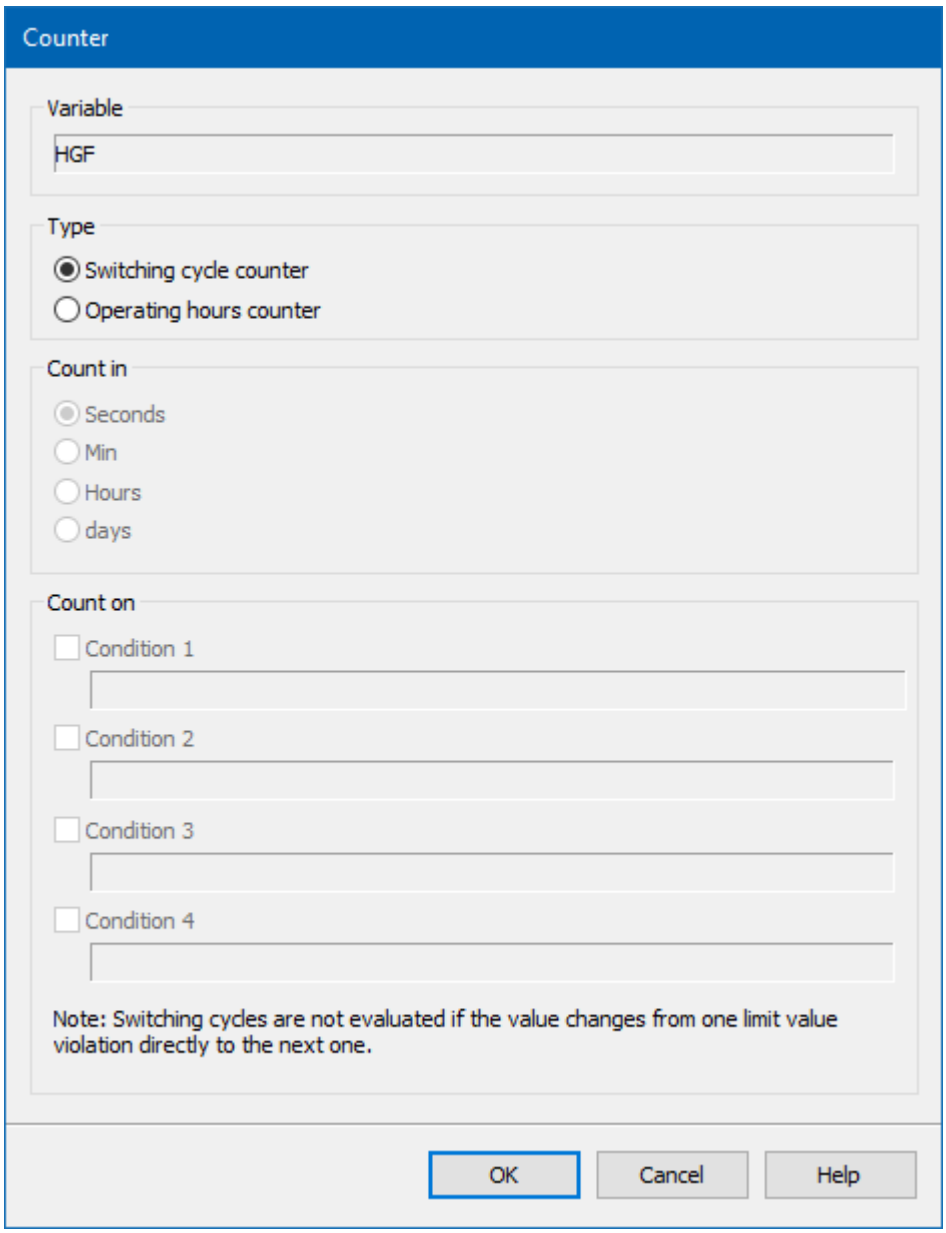
Attention

If a counter variable - with the formula: *Count()* - of the mathematics driver is saved in an archive for which *saving is carried out in the event of a value change* (spontaneous), this can lead to very large amounts of data being saved.

Workaround: Assign the mathematics variable of a simulation variable to the (**SIMUL32** driver, *HD* **driver object type**) and save the simulation variable in the archive.
Attention: The simulation variable must be an integer data type (e.g. *INT*, *DINT*), and not *REAL*.

CONFIGURING THE COUNTER

By clicking on OK in the statistics dialog, the dialog to configure the counter type and the states or limit values to be counted is opened:



VARIABLE

Display of the variables to be configured.

TYPE

Counter type. Selection between switching cycle counter and operating hours counter:

Option	Description
Switching cycle counter	Counts if a state or limit value has been reached after a delay time that may have been configured has expired. **Note:** Switching cycles are not evaluated if the value switches from one limit value violation directly to another limit value violation. For example, limit value violations of binary variables cannot be evaluated with the **switching cycle counter**.
Operating hours counter	Counts the time as long as the variable has the selected state or limit value. The time of any possible failure of communication is taken into account. **Example:** assume that the counter is supposed to count the operating hours of value 1. When the last value is *1* and ▸ the communication goes to *failed* (variable has *INVALID* status bit) or ▸ Runtime is closed the counter will continue to count. The counter will stop as soon as the communication is restored and the value is *0*.

COUNT IN

Activation of the states in which counting is to take place.

Option	Description
Count in	If the source variable is linked to a reaction matrix, the states 1 - 4 match the counter numbers of the Rema. If the source variable was not linked to a reaction matrix, the possible conditions correspond to the limit values.

CLOSE DIALOG

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

COUNTING METHOD

The counter only counts new violations of the sum of all configured limit values, but not violations of individual values. That means:

- The counter value is not increased if:
  - A limit value has already been violated and
  - The variable value changes so that a limit value is violated again
- The counter value is not increased if:
  - The variable value changes within a limit value and
  - The **Treat each value change as a new limit value violation** property is activated
- The counter value is only increased if:
  - None of the 4 states was previously violated
  - At least one of the 4 states after the update was violated

7.1.4.5 Cross calculation

The cross calculation allows the calculation of mean values over several variables.

Note: Cross calculations cannot be combined with any further functions.

Select **Cross calculation** from the drop-down list in the **Formula definition** section. The following basic functions are available:

Expression	Function
CrossMean	Calculation of the average of several channels at the same time.
()	Brackets

Attention: Calculations with the function **CrossMean()** always use all source variables selected in the mathematics variable independently on the selected parameters.
Example: The list contains 5 variables, 3 are selected. The result of the calculation with **CrossMean()** relates to all 5 existing variables.

7.1.4.6 Data reduction

Select **Data reduction** from the drop-down list in the **Formula definition** section. The following basic functions are available:

Expression	Function
TMittelung	Average over time and events (on page 51)
Sum	Sum over time and events (on page 54)
Max	Maximum over time and events (on page 57)
Min	Minimum over time and events (on page 59)
Dif	Difference over time (on page 61)
()	Brackets

Data reduction cannot be combined with any further functions.

NOTES ON CALCULATION

SUM AND CHRONOLOGICAL AVERAGE VALUE

The values are calculated and shown at the end of the interval. The end is determined using the options **time limited** or **event-triggered**. If the **continuous** option is selected, the display depends on the sampling behavior of the source variable. The sampling behavior corresponds to the new value or the sampling rate. The sampling rate is defined as 0.5 seconds or event-triggered. The average value is formed from the sum through the corresponding division (the number of value changes, for example).

Options:

- **Initialization at the interval beginning**:

  The value from the end of the interval remains displayed.

  - **With current value** option active: The last measured value becomes the new sum. After the first value is received, the average is formed from these two values - and so forth.
  - **With new value** option active: The sum starts with 0. Only the newly-received values are used to form the average. If no new values are received for an interval, the variable for the average value will be set to invalid (**I_BIT**).

- **Calculate as:**
  - **Integral over the time**: Is a time-weighted sum. The respective previous value is added up and multiplied by the duration of the sampling period in seconds, according to the sampling behavior. The average is formed from division by the interval duration.

Behavior when setting values:

- If a set value is written, this has the effect of a restart of the interval. The set value must not be *0* for this.
- The set value is used as a current sum for **with current value**.
- For **with new value**, the sum is set to *0*. The average is calculated. The canceled interval is evaluated. If **output at the end of the interval** is selected, this value remains in the display. For continuous **output**, the current sum or the average appears immediately.

MAXIMUM AND MINIMUM

The display depends on the selected options for output and initialization.
Options:

- **Output of the result**:
  The extreme value is calculated each time a value changes an output depending on the option selected:
  - Option **At the end of the interval**: The extreme value is output at the end.
  - **Continuous** option: The extreme value is shown as updated on a continuous basis.
- **Initialization at the start of the interval**:
  Reference for the next display:
  - Option **With current value**: The last measured value becomes the current extreme value.
  - Option **With new value**: The extreme value is deleted. If no new values are received during an interval, the extreme value variable is invalid (**I_BIT**).

Behavior when setting values:

- The set value becomes the new extreme value, regardless of the option selected for **Initialization at the start of the interval**.
- The value is shown immediately with the **Continuous** option.

7.1.4.6.1 Time average

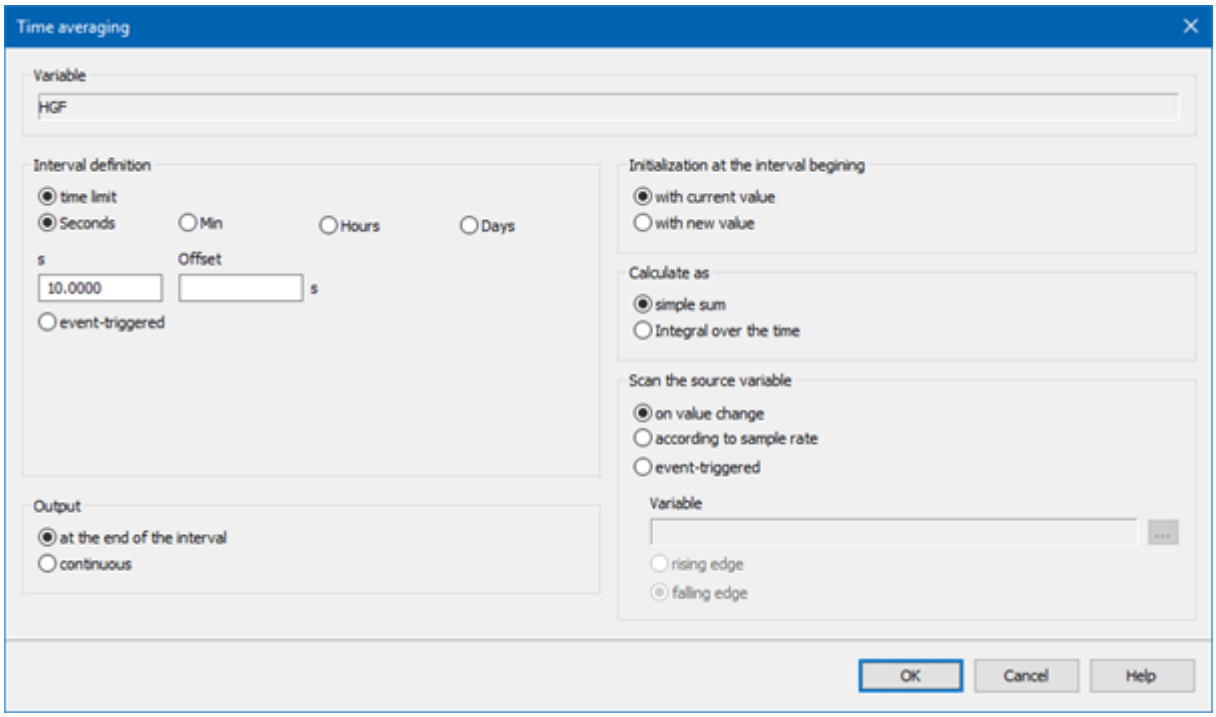
The data reduction **Time average** allows the calculation of the average of a variable. The possible average calculations are:

- Time interval:
  Averaging over time interval (seconds, minutes, hours, days)
- Event interval:
  The average is calculated when a change in an event variable occurs

Further parameters in the configuration are:

- Output: The result is provided at the end of the interval or continuously
- The value is initialized at the beginning of the interval (existing or new value)
- Type of calculation: Stipulation of the calculation type (sum, integral via the time)
- Scanning type: Type of scanning of the source Variable (spontaneous, after update parameterization, event-controlled)

Press the button **OK** to open a dialog where you can enter the parameters for calculating the average:



VARIABLE

Display of the event variable.

INTERVAL DEFINITION

Option	Description
time limit	Calculation of average over time.

Option	Description
event-triggered	Average is calculated when an event variable changes.
s	Entry of the numeric value for the corresponding time (real-time synchronous, i.e. entry 15 means 15, 30, 45, 0 etc.).
Seconds, minutes, hours, days	Selection of the time interval.
Offset	Delay for start and end of the average calculation.

Output

Parameter	Description
At the end of the interval	The value is only provided at the end of the interval; until then, only the previous value is available.
continuous	The value is updated during each scan of the source variable.

INITIALIZATION AT THE INTERVAL BEGINING

Option	Description
with current value	Value is based on previous value.
with new value	Value must be re-calculated.

Calculate as

Parameter	Description
simple sum	Addition of the values.
Integral over the time	Integration of the values over the time interval

SCAN THE SOURCE VARIABLE

Configuration of the source variable using the following options:

Option	Description
on value change	Spontaneous changes of the source variables are immediately taken into account for the calculation
according to sample rate	The source values are only used in the defined update rate for the calculation (button **Update is fixed to 0,5 sec.**).
event-triggered	Source value is used for the calculation when a change of

Option	Description
	edge occurs in the configured binary event variable.
Variable	Selection of the binary event variable. Click on the **...** button to open the selection dialog. Determination of the respective edge change for the acceptance of the source value using radio buttons. ▸ **rising edge** ▸ **falling edge** Only available if the **event triggered** property is active.

7.1.4.6.2 Summation

The data reduction **Summation** allows to calculate the sum of a variable.

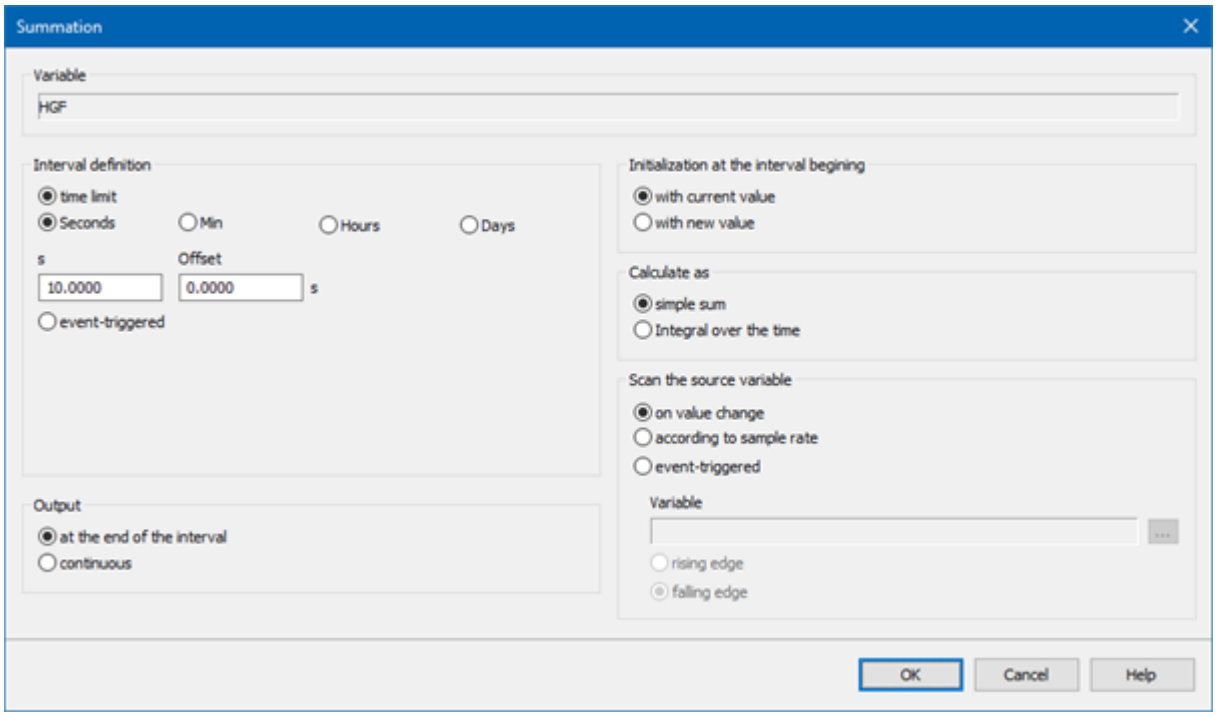
Possible summations are:

- Time interval:
  Averaging over time interval (seconds, minutes, hours, days)
- Event interval:
  The sum is formed when a change to an event variable occurs

Further parameters in the configuration are:

- Output: The result is provided at the end of the interval or continuously
- The value is initialized at the beginning of the interval (existing or new value)
- Type of calculation: Stipulation of the calculation type (sum, integral via the time)
- Scanning type: Type of scanning of the source Variable (spontaneous, after update parameterization, event-controlled)

Press the button **OK** to open a dialog where you can enter the parameters for calculating the sum:



Variable
Display of the event variable

INTERVAL DEFINITION

Option	Description
time limit	Minimum over time.
event-triggered	Minimum is calculated when an event variable changes.
s	Entry of the numeric value for the corresponding time (real-time synchronous, i.e. entry 15 means 15, 30, 45, 0 etc.).
Seconds, minutes, hours, days	Selection of the time interval.
Offset	Delay for start and end of the average calculation.

OUTPUT

Option	Description
At the end of the interval	The value is only provided at the end of the interval; until then, only the previous value is available.

Option	Description
continuous	The value is updated during each scan of the source variable.

INITIALIZATION AT THE INTERVAL BEGINING

Option	Description
with current value	Value is based on previous value.
with new value	Value must be re-calculated.

CALCULATE AS

Option	Description
simple sum	Addition of the values.
Integral over the time	Integration of the values over the time interval

SCAN THE SOURCE VARIABLE

Configuration of the source variable using the following options:

Option	Description
on value change	Spontaneous changes of the source variables are immediately taken into account for the calculation
according to sample rate	The source values are only used in the defined update rate for the calculation (button **Update is fixed to 0,5 sec.**).
event-triggered	Source value is used for the calculation when a change of edge occurs in the configured binary event variable.
Variable	Selection of the binary event variable. Click on the **...** button to open the selection dialog. Determination of the respective edge change for the acceptance of the source value using radio buttons. ▸ **rising edge** ▸ **falling edge**

Option	Description
	Only available if the **event triggered** property is active.

7.1.4.6.3 Maximum calculation

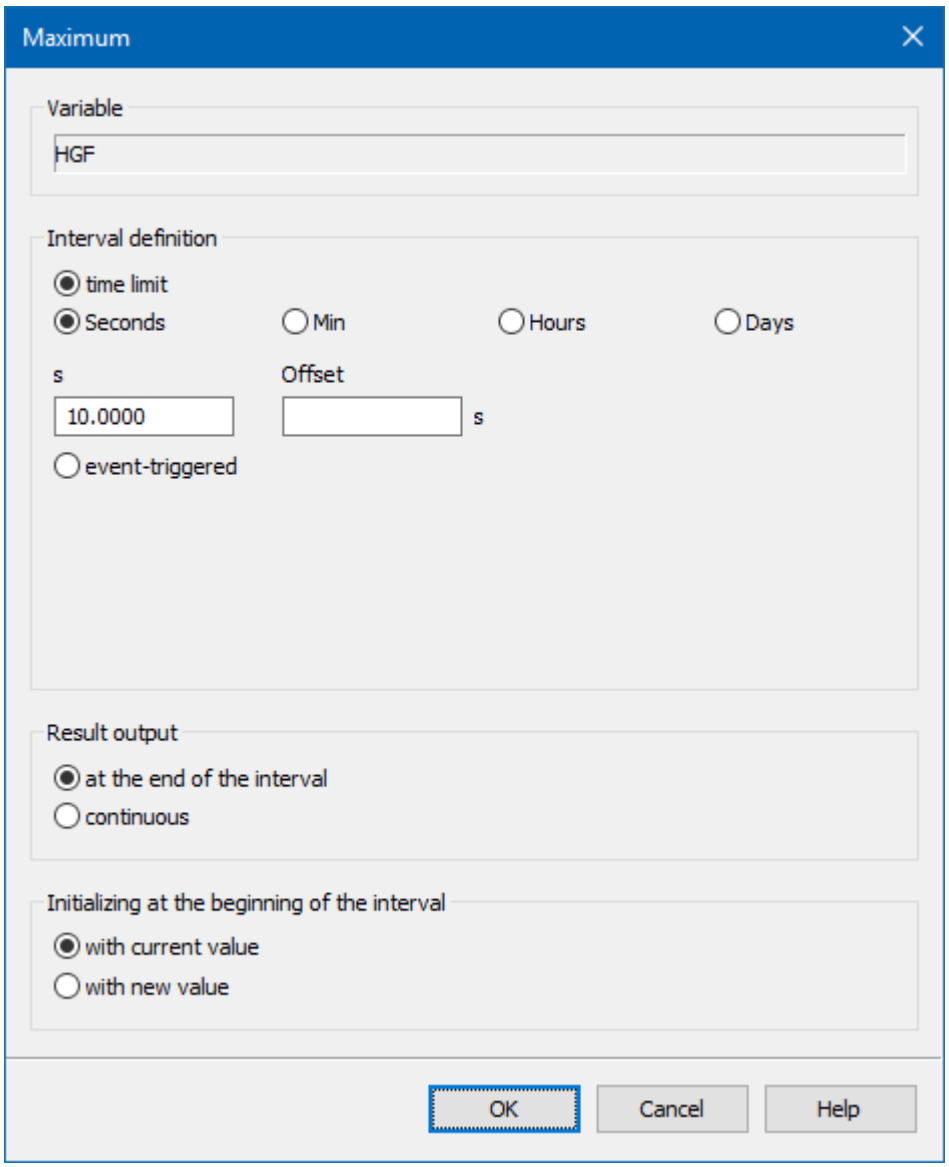
The data reduction of maximum offers the possibility to determine the maximum of a variable within the interval. Possible settings are:

- Time interval:
  Maximum within a time interval (seconds, minutes, hours, days)
- Event interval:
  The maximum is formed when a change in an event variable occurs

Further parameters in the configuration are:

- Output: The result is provided at the end of the interval or continuously
- The value is initialized at the beginning of the interval (existing or new value)

Press the button OK to open a dialog where you can enter the parameters for calculating the maximum:



Variable: Display of the event variable

INTERVAL DEFINITION

Option	Description
time limit	Maximum over time.
Seconds, minutes, hours, days	Selection of the time interval.
s	Entry of the numeric value for the corresponding time (real-time synchronous, i.e. entry 15 means 15, 30, 45, 0

Option	Description
	etc.).
Offset	Delay for start and end of the calculation of the maximum.
event-triggered	Maximum is calculated when an event variable changes.

RESULT OUTPUT

Option	Description
At the end of the interval	The value is only provided at the end of the interval; until then, only the previous value is available.
continuous	The value is updated during each scan of the source variable.

INITIALIZING AT THE BEGINNING OF THE INTERVAL

Option	Description
with current value	Value is based on previous value.
with new value	Value must be re-calculated.

> **Information**
>
> The synchronization time is determined based on the time interval.
>
> Example: If you configure the time interval to be 15 minutes, the calculation will be performed every 15 minutes, starting with the full hour.

7.1.4.6.4Minimum calculation

The data reduction of minimum offers the possibility to determine the minimum of a variable within the interval. Possible settings are:

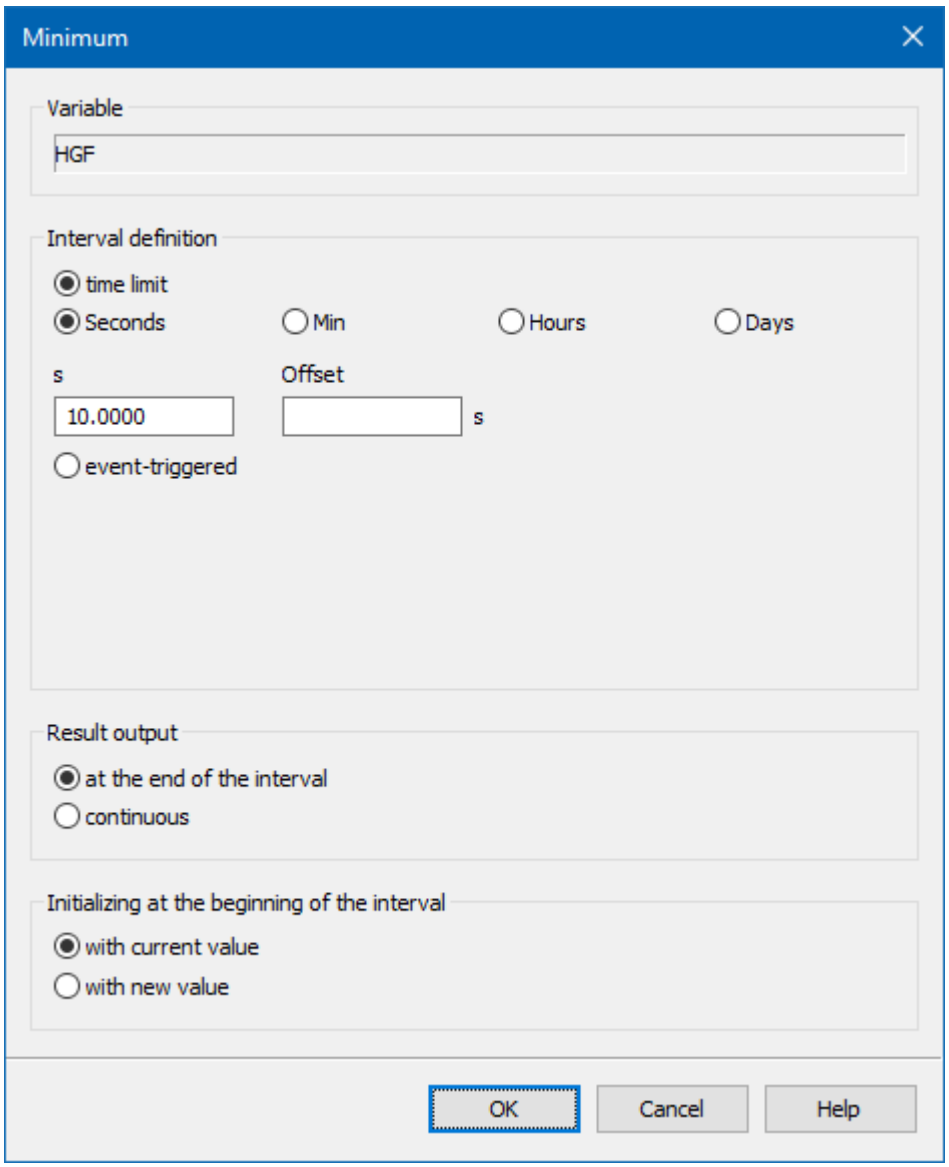
- Time interval: Minimum within a time interval (seconds, minutes, hours, days)
- Event interval: The minimum is formed when a change in an event variable occurs

Further parameters in the configuration are:

- Output: The result is provided at the end of the interval or continuously

- The value is initialized at the beginning of the interval (existing or new value)

Press the button OK to open a dialog where you can enter the parameters for calculating the minimum:



Variable: Display of the event variable

INTERVAL DURATION

Option	Description
Time limit	Minimum over time.
Seconds, minutes, hours, days	Selection of the time interval.
s	Entry of the numeric value for the corresponding time

Option	Description
	(real-time synchronous, i.e. entry 15 means 15, 30, 45, 0 etc.).
Offset	Delay for start and end of the calculation of the minimum.
event-triggered	Minimum is calculated when an event variable changes.

RESULT OUTPUT

Option	Description
at the end of the interval	The value is only provided at the end of the interval; until then, only the previous value is available.
continuously	The value is updated during each scan of the source variable.

INITIALIZING AT THE BEGINNING OF THE INTERVAL

Option	Description
with current value	Value is based on previous value.
with new value	Value must be re-calculated.

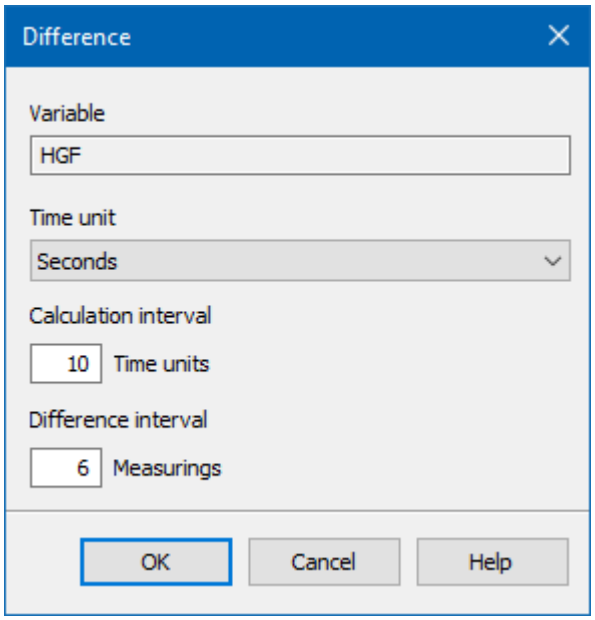
> **Information**
>
> The synchronization time is determined based on the time interval.
>
> Example: If you configure the time interval to be 15 minutes, the calculation will be performed every 15 minutes, starting with the full hour.

7.1.4.6.5 Difference over time

The function can be used e.g. for dynamically monitoring the consumption of liquids or for checking containers for leakages (based on fill level values).

The variable that was inserted into the function is monitored and its values are stored internally with the configurable measurement distance (in time units). If a specific difference is reached (number of measurements), the function will display the first result (R(n)-R(0)), which will then be updated after each measurement. The result of the function is an element of the set of real numbers.

Press the button **OK** to open a dialog where you can enter the parameters for calculating differences:



Option	Description
Variable	Display of the event variable.
Time unit	Drop-down list for selecting the time unit (seconds, minutes, hours, days)
Calculation interval	
Time units	Number of units for the calculation interval.
Difference interval	
Measurings	Number of measurements for the difference interval.

Information

The calculation of the difference over time is not absolute but relative, starting with the Runtime start.

7.1.4.7 Comparison

Select **Comparison** from the drop-down list in the formula definition section. The following operators are available:

Expression	Function
<	less than
>	greater than
<=	Less than or equal
>=	greater or equal
<>	less or greater
=	equal
()	Brackets

Comparison operators can be combined with:

- Arithmetics (on page 41)
- Trigonometry (on page 42)
- Boolean algebra (on page 42)

7.1.4.8 Counting value processing

For linkages with time or event reference (statistics, data reduction) it is necessary to make additional entries. You can make them when ending the link.

Attention

This function is only available for SICAM 230 and requires a corresponding license. Find out more in the help pages of SICAM 230.

7.1.4.9 Additional float formulas

You can use further float formulas that cannot be reached via buttons. You have to enter them directly in the area **Formulas**:

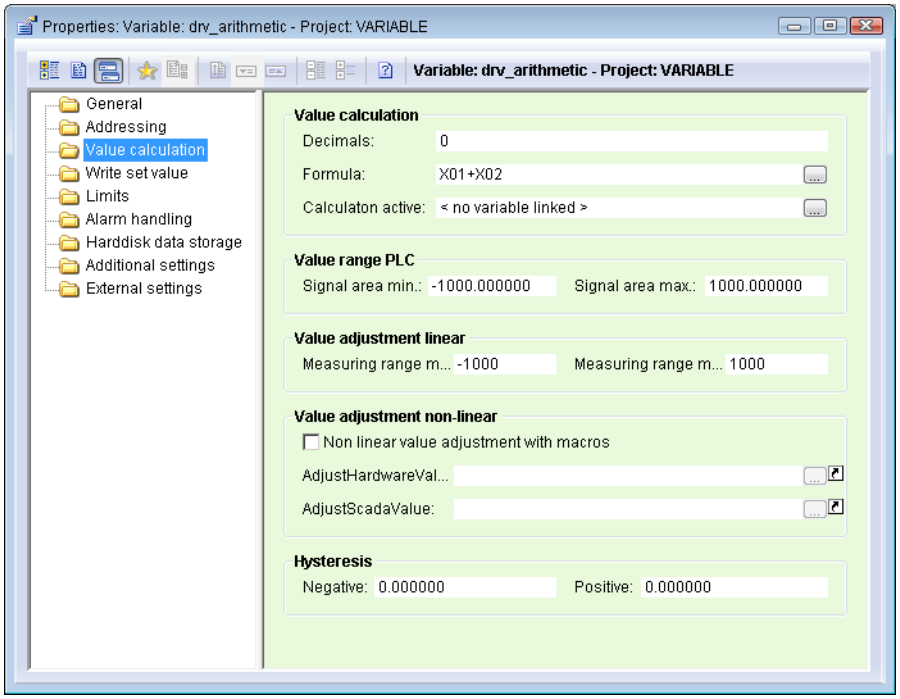
Syntax	Description	Notation
ABS(X01)	Absolute value	|X01|
EXP(X01)	Exponent	$e^{(X01)}$
LN(X01)	Natural logarithm	ln(X01)

Syntax	Description	Notation
LOG(X01)	Logarithm	log(X01)
PI*(X01)	Constant Pi	PI*(X01)
SQR(X01)	Square calculation	$(X01)^{2}$
SQRT(X01)	Square root calculation	$(X01)^{1/2}$

7.2 Event-triggered calculation of Mathematics variables

A Mathematics variable can be assigned to a variable, whose value triggers a calculation. In order to assign a variable for the activation of the calculation:

1. navigate to the **Value calculation** node in properties
2. in the property **Calculaton active**, click on the field with the variable name or on the button **...**
3. the dialog for assigning variables opens now
4. select the variable you want to assign



FUNCTIONALITY

- Mathematics variables without an activation variable register all source variables, interval variables and scan variables when the Runtime starts.
- Start of the calculation:
  If a Mathematics variable is linked to an activation variable, the calculation starts as soon as the activation variable has a value of <> *0*.
  - If the activation variable changes to *activated*, all source variables, interval variables and scan variables will be requested.
  - If the activation variable changes back to *deactivated*, they will be signed off again.
- The activation has priority over the calculation.

  This shows when the activation variable serves as a source variable for the Mathematics variable. If the activation variable changes to deactivated, the result will stay at the same value as it was at the time when the calculation was activated.
- The data type *String* is allowed if the value can be converted to a number.

  If the conversion fails (empty, no number), this will result in the status *deactivated*.
- Status of the Mathematics variables:
  - *Deactivated:S_AUS* (Bit 20, switched off).
    The current value of the Mathematics variable will be used as the value. If no value exists yet, the replacement value of the Mathematics variable will be used. Mathematics variables retain the status *S_AUS* until there is a calculation result, e. g. calculation type **Counter -> Switching cycle counter with status**.
  - for the calculation types *Min* and *Max*, with an interval variable but without an activation variable, until the edge of the interval variable is received: *S_IBIT* (Bit 18, invalid)
  - for the calculation types *Min* and *Max*, with an interval variable and an activation variable, until the edge of the interval variable is received: *S_AUS*
- For Mathematics variables that have an interval variable or a scan variable, edge recognition works only if calculation is activated.

  If the interval variable or the scan variable is already active when the activation variable changes from deactivated to activated, this corresponds to an edge of the interval variable / the scan variable.
- If you write to a Mathematics variable, whose calculation is deactivated:
  - the written value will not be sent and is therefore not visible.
  - The written value serves as the initial value (current value) of the calculation, as soon as the calculation is activated. **However:**
  - The write command will be sent to the Standby Server, independent of the activation variable.

7.2.1 Activation and deactivation of the calculation

ACTIVATION

Calculation will be activated if the following conditions apply for the activation variable after a value change:

- The activation variable has a value
- The value changes from *0* to <> *0* and one of the following states is active:
  - *S_GA* General interrogation Bit 16
  - *S_SPONTAN* Spontaneous Bit 17
  - *S_REVISION* Revision Bit 9
  - *S_EW_KENNUNG* Replacement value Bit 27

This allows you to trigger a calculation by switching to a replacement value.

DEACTIVATION

The calculation will be deactivated if the following conditions apply after a value change of the activation value:

- The variable does not have a valid value yet.
- The value is *0* or none of the states *S_GA*, *S_SPONTAN*, *S_REVISION* or *S_EW_KENNUNG* is active.

7.2.2 Calculation types with special behavior

COUNTER: COUNT WITH STATE

If the caluclation for **Count with state** is deactivated, neither the receiving nor the clearing of the state will be recognized.

This means: If the activation variable is *deactivated*, the state is considered as *not violated*.

For example:

1. Status received
2. Activation variable changes to *activated*

3. **Violation is recognized -> Counter gets a value and counts**
4. Activation variable *deactivated*
5. Status cleared
6. Status received
7. **Activation variable activated -> no counting**
8. Status cleared
9. **Status received -> counting**
10. Activation variable *deactivated*
11. Status cleared
12. Activation variable *activated*
13. **Status received -> counting**

7.2.3 Error messages

Error message	Reason	Solution
Source variable <*VariablenProjekt Name*>/<*Variablen-ID*> for mathematics missing. Mathematics variable <*Projektname*>/<*MaV Name*> disabled!	The source variable of the mentioned project is missing. The Mathematics variable is not calculated.	Change configuration or start project.
Advise for source variable <*VariablenProjekt Name*>/<*Variable Name*> failed. Mathematics variable <*Projektname*>/<*MaV Name*> disabled	Data point request for the variable of the mentioned project has failed. The Mathematics variable is not calculated.	Increase system resources, restart Runtime.

7.3 Importing Mathematics variables

When importing Mathematics variables (via XML import), make sure that all variables used in the formulas already exist. If you import variables of different drivers, we recommend to perform the import a second time. This makes sure that recently imported variables are also linked to the Mathematics driver formulas.

> **Attention**
>
> Network: On a Standard Server defined as data server, the Mathematics variables are not displayed if it is upgraded to be a server.

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.

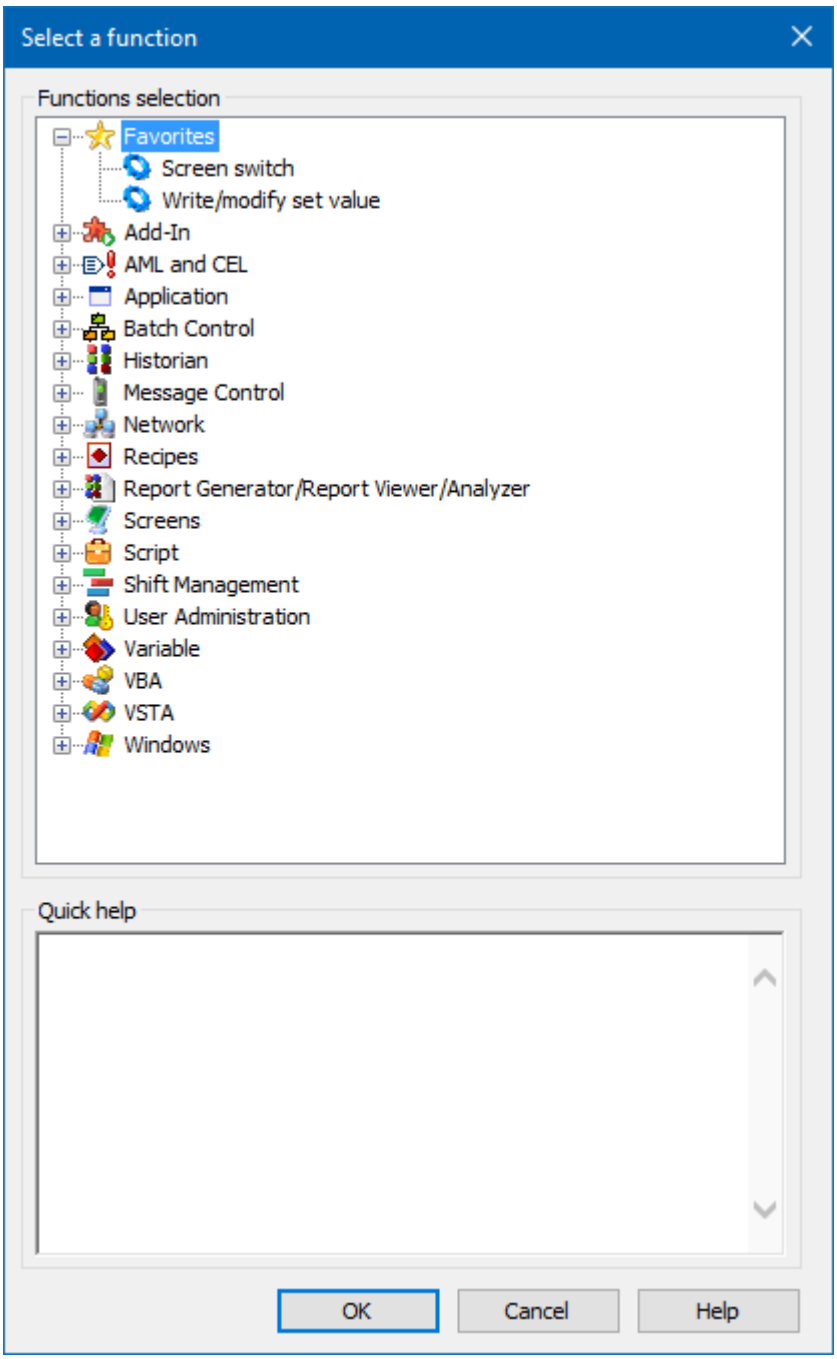
> **Attention**
>
> The zenon **Driver commands** function is not identical to driver commands that can be executed in the Runtime with Energy drivers!

CONFIGURATION OF THE FUNCTION

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

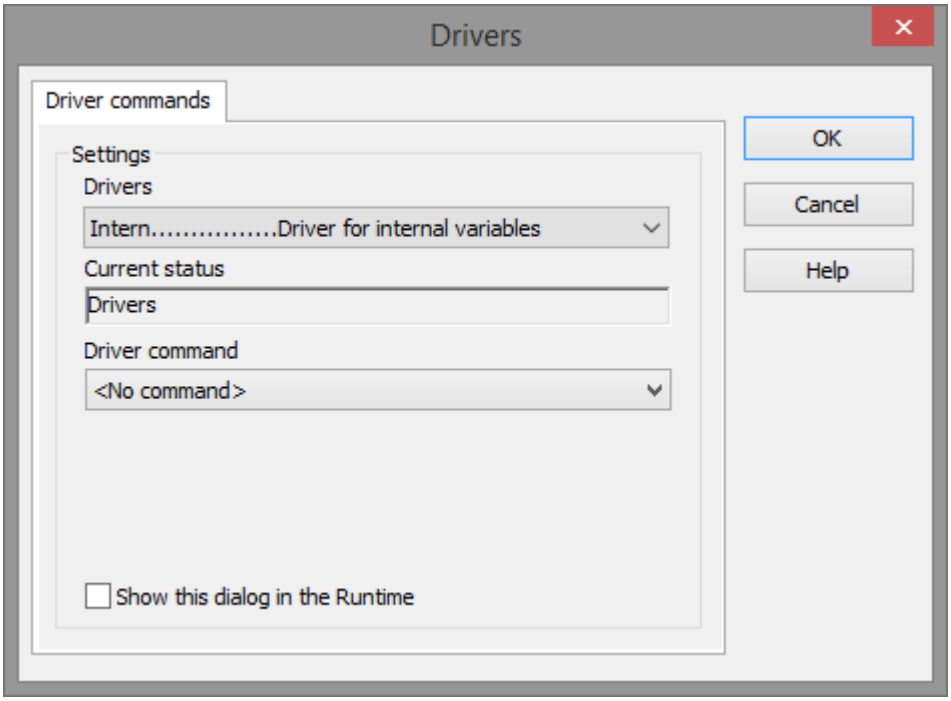
1. Create a new function in the zenon Editor.

The dialog for selecting a function is opened



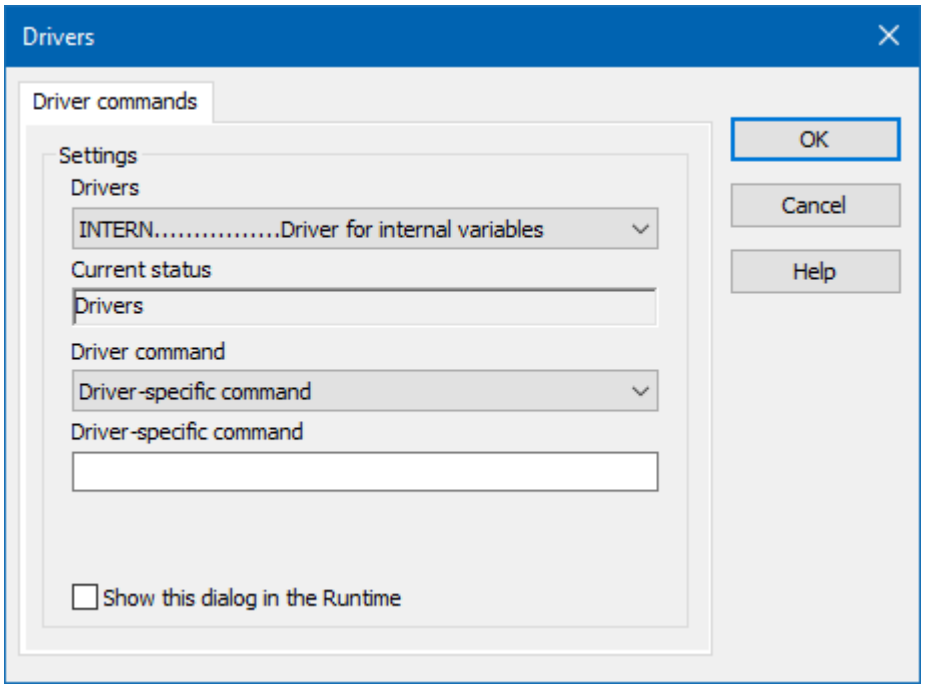
2. Navigate to the node **Variable.**
3. Select the **Driver commands** entry.

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.

9 Error analysis

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

9.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer program that was also installed with zenon. You can find it under **Start/All programs/zenon/Tools 8.20 -> Diagviewer.**

zenon driver log all errors in the LOG files.LOG files are text files with a special structure. The default folder for the LOG files is subfolder **LOG** in the folder **ProgramData**. For example:

%ProgramData%\COPA-DATA\LOG.

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

In the Diagnosis Viewer you can also:

- Follow newly-created entries in real time
- customize the logging settings
- change the folder in which the LOG files are saved

Note:

1. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.

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

> **Attention**
>
> In Windows CE errors are not logged per default due to performance reasons.

You can find further information on the Diagnosis Viewer in the Diagnose Viewer manual.

9.2 Driver monitoring

Runtime monitors the availability of the driver by means of a watchdog. If a driver is no longer available, the *INVALID* status bit is also set for all checked-in variables.

Possible causes for a triggering of the watchdog:

- The driver process is no longer running.

  Check whether the driver EXE file is still running in the Task Manager.
- Operating system is busy with processes that have a higher priority.

  Check the configuration of your system to see whether there is sufficient memory and CPU power. In this case, the driver only resets the *INVALID* status bit if there is a value change on the connected party. Static values retain the *INVALID* status bit until the next time the Runtime or the driver is started.

CONFIGURATION OF WATCHDOG

For the monitoring of communication in the Runtime, the connection to the driver is checked in a fixed, prescribed time period of 60 seconds. This process is repeated several times. If, within 5 attempts (= within 5 minutes), no valid connection to the driver is detected, the *INVALID* bit is set for the checked-in (*advised*) variables. In addition, the *INVALID* bit is also set when new variables are advised. The *INVALID* bit will no longer be reset.

Corresponding LOG entries are created for this.

LOG ENTRY

An error message is logged in the LOG when the watchdog is triggered:

Parameter	Description
Communication with driver:<drvExe>/<drvDesc>(id:<drvId>) timed out. No communication for <time> ms.	No communication with driver within the given time. ▸ *<time>*: Time (in milliseconds) ▸ *<drvDesc>*: Driver name ▸ *<drvExe>*: Driver EXE name ▸ *<drvId>*: Driver ID in the zenon project
Communication with %s timed out. Invalid-Bit will be set.	Communication to the *%s* driver could not be established after 5 attempts within 60 seconds. The *INVALID* bit is set for the variable.
Communication with %s timed out. Timeout happened %d times	Communication to the *%s* driver could not be established after *%d* times within 60 seconds.

9.3 Log server

All messages and logs of the driver are sent to the Log Server. The messages can be displayed with the Diagnosis Viewer. (Older versions of zenon use an "Error text file".

By default, the messages of the type 'Error' for the module driver (DRV) will be logged. If you want extended logs to be created, you have to configure this accordingly in the client settings of the Diagnosis Viewer in the Runtime: **Settings – Client configuration, List of parameters, ' Configuration of the message level** '; confirm with **Accept** .

9.4 Check list

- Analysis with the Diagnosis Viewer (on page 73):
  -> Which messages are displayed?