



© 2020 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

Distribution and/or reproduction of this document or parts thereof in any form are permitted solely with the written permission of the company COPA-DATA. Technical data is only used for product description and are not guaranteed properties in the legal sense. Subject to change, technical or otherwise.



# **Contents**

1	Welcome to COPA-DATA help	4
2	CAN	4
3	CAN - data sheet	5
4	Driver history	6
5	Requirements	6
	5.1 PC	6
	5.2 PLC	7
6	Configuration	7
	6.1 Creating a driver	8
	6.2 Settings in the driver dialog	
	6.2.1 General	
	6.2.2 Options	
7	Creating variables	17
	7.1 Creating variables in the Editor	17
	7.2 Addressing	21
	7.3 Driver objects and datatypes	22
	7.3.1 Driver objects	
	7.3.2 Mapping of the data types	23
	7.4 Creating variables by importing	24
	7.4.1 XML import	24
	7.4.2 DBF Import/Export	25
	7.5 Communication details (Driver variables)	31
8	Driver-specific functions	36
9	Driver command function	37
10	) Error analysis	42
	10.1 Analysis tool	42
	10.2 Driver monitoring	
	10.3 Check list	



# 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 CAN

The CAN driver communicates with the serial CAN bus system (Controller Area Network).

- ▶ The connection from the computer is established by means of a USB port and a *Lawicel CAN USB Converter*.
- ▶ The variables are addressed with numerical addressing.



# 3 CAN - data sheet

General:	
Driver file name	CAN.exe
Driver name	Generic driver for CAN Bus
PLC types	Standard CAN communication
PLC manufacturer	CAN; Lawicel

Driver supports:	
Protocol	CAN
Addressing: Address-based	Address based
Addressing: Name-based	
Spontaneous communication	
Polling communication	X
Online browsing	
Offline browsing	
Real-time capable	
Blockwrite	
Modem capable	
RDA numerical	
RDA String	
Hysteresis	
extended API	
Supports status bit WR-SUC	
alternative IP address	



Requirements:	
Hardware PC	CAN Converter CANUSB from Lawicel
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	Build number	Change
15.12.20 <sup>-</sup> 5	24560	Created driver documentation

# 5 Requirements

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

### 5.1 PC

The following requirements must be met for **CAN driver** communication:

- A free USB interface must be available on the PC.
- ▶ A Lawicel CAN USB converter must be connected to a free USB 2.0 or USB 3.0 interface.



▶ The appropriate software for the *Lawicel CAN USB* converter must be installed on the computer.

**Note:** You can find downloads on the homepage, www.canusb.com (www.canusb.com)

### Hint

The hardware configuration can be tested with the **CAN Monitor PRO** software.

The CAN Monitor Pro Program was developed by WGSoft.de and makes it possible, with the help of a LAWICEL CANUSB interface adapter, to carry out tests and send messages to the CAN bus in a simple manner.

CAN frames with desired parameters (CAN ID, data length, speed, etc.) can thus be sent, received and logged

### 5.2 PLC

The controller can support the CAN bus system and be connected to the bus.

# 6 Configuration

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

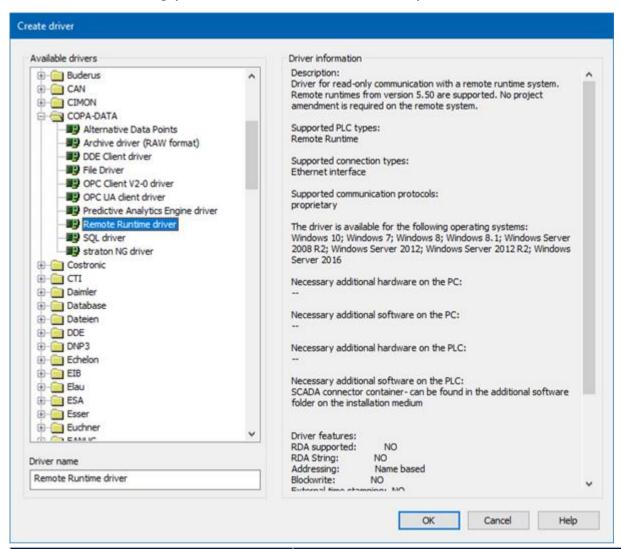
# Information

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



# 6.1 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
	Default. No selection
Driver name	Unique <b>Identification</b> of the driver.
	Default: empty
	The input field is pre-filled with the pre-defined



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

#### **CLOSE DIALOG**

Option	Description
ОК	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:

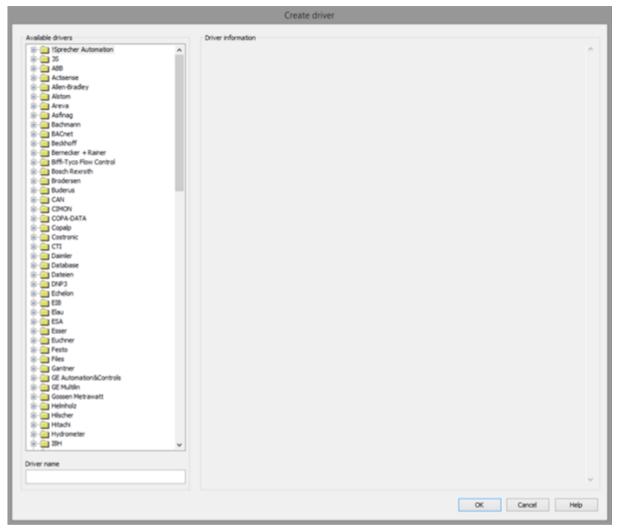
1. Right-click on **Driver** in the Project Manager and select **New driver** in the context menu.

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.

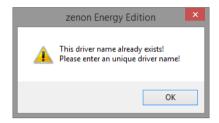
  Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (\_).
- **Attention:** This name cannot be changed later on.
- Confirm the dialog by clicking on the OK button.
   The configuration dialog for the selected driver is 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



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

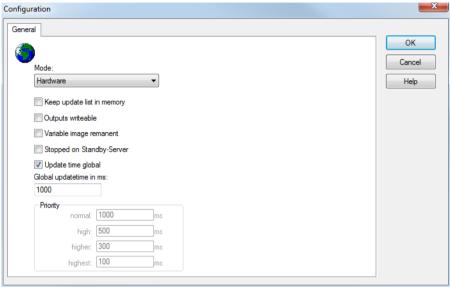
## 6.2 Settings in the driver dialog

You can change the following settings of the driver:



### 6.2.1 General

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
	<ul><li>Hardware:</li><li>A connection to the control is established.</li></ul>
	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.
	<ul> <li>Simulation - counting:         <ul> <li>No communication between to the control is established, the values are simulated by the driver.</li> <li>In this modus the driver increments the values within a value range automatically.</li> </ul> </li> </ul>
	<ul> <li>Simulation - programmed:</li> <li>No communication is established to the PLC. The</li> </ul>



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.
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	<ul><li>Active: Outputs can be written.</li></ul>
	<ul><li>Inactive: Writing of outputs is prevented.</li></ul>
	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 <i>M1</i> (0) to <i>M8</i> (7)
	► REVISION(9)
	► AUS(20)
	► ERSATZWERT(27)
	The variable image is always saved if:
	the variable is of the Communication details object type
	<ul> <li>the driver runs in simulation mode. (not programmed simulation)</li> </ul>
	The following states are not restored at the start of the Runtime:



Option	Description
	► SELECT(8)
	▶ WR-ACK(40)
	► WR-SUC(41)
	The mode <b>Simulation - programmed</b> 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.
	<b>Attention:</b> 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
	<b>Note:</b> 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:
	<ul> <li>Active:         The set Global update time is used for all variables in the project. The priority set at the variables is not used.     </li> </ul>
	<ul> <li>Inactive:         The set priorities are used for the individual variables.     </li> </ul>
	<b>Exceptions:</b> Spontaneous drivers ignore this option. They generally use the shortest possible update time. For details, see the <b>Spontaneous driver update time</b> section.



Option	Description
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.
	<b>Attention:</b> Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.

#### **CLOSE DIALOG**

Option	Description
ОК	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**

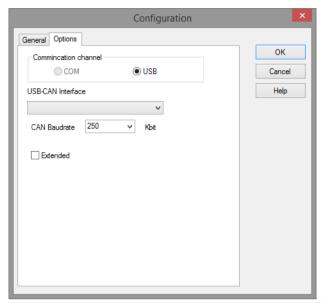
With spontaneous drivers, for **Set value**, **advising** of variables and **Requests**, a read cycle is triggered immediately - regardless of the set update time. This ensures that the value is immediately available for visualization after writing. The update time is generally 100 ms.

Spontaneous drivers are ArchDrv, BiffiDCM, BrTcp32, DNP3, Esser32, FipDrv32, FpcDrv32, IEC850, IEC870, IEC870\_103, Otis, RTK9000, S7DCOS, SAIA\_Slave, STRATON32 and Trend32.



# 6.2.2 Options

You configure the communication parameters in this tab.



Parameter	Description
Communication channel	<ul> <li>Option field to select the communication channel:</li> <li>USB         communication via USB interface.</li> <li>COM         Communication via serial interface.</li> <li>Note: This option cannot currently be selected and is always grayed out.</li> </ul>
USB-CAN Interface	Drop-down list for the selection of the USB interface.  All USB converters that are connected and have been detected on the computer are shown in the list.
CAN Baudrate	Drop-down list for the selection of the transfer rate.  • 10 - 1000 Kbit  Default: 250 Kbit  Note: This CAN Baud rate must match the one set at the CAN bus.



Parameter	Description
Extended	Checkbox for the selection of the communication format.
	• active:
	Data telegrams are read and written in Extended Frame Format (2B, 29-bit ID).
	• inactive:
	Data telegrams are read and written in Base Frame Format (2A, 11-bit ID).
	Default:nicht aktiv

# 7 Creating variables

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

# 7.1 Creating variables in the Editor

Variables can be created:

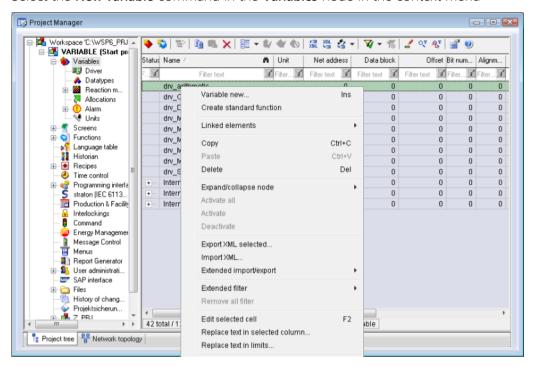
- as simple variables
- in arrays
- as structure variables

### **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.	
	<b>Note:</b> If no driver has been opened in the project, the driver for internal variables ( <b>Intern.exe</b> ) 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
- POPCUA32
- 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.

# 7.2 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	The CAN message ID of the message block is in the data block field	
Offset	Byte offset of the variable within the CAN message block. This has a length of 8 bytes.	
Alignment	not used for this driver	
Bit number	Number of the bit within the configured offset.	
	Possible entries: 0 7	
String length	Only available for String variables.  Maximum number of characters that the variable can take.	
Driver connection/Driver Object Type	Is only available for "MessageID".	
Driver connection/Data	Data type of the variable. Is selected during the creation of the variable; the type can be changed here.	
Туре	<b>Attention:</b> If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.	



# 7.3 Driver objects and datatypes

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

# 7.3.1 Driver objects

The following object types are available in this driver:

Driver Object Type	Channe I type	Read	Write	Supported data types	Description
MessageID	34	X	X**	BOOL, SINT, UINT, LREAL, DINT, LWORD, INT, ULINT*,REAL, LINT*,DATE, UDINT, USINT, TOD, Long double	*Caution: In zenon, values are always displayed in LREAL format after being rounded up.  **Writing is always only for one byte according to the set offset (0-7).
Communication details	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Variables for the static analysis of the communication; Values are transferred between driver and Runtime (not to the PLC).
					<b>Note</b> : The addressing and the behavior is the same for most zenon drivers.
					You can find detailed information on this in the Communication details (Driver variables) (on page 31) chapter.

### Key:

**X**: supported



--: not supported

#### **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.

# 7.3.2 Mapping of the data types

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

PLC	zenon	Data type
BOOL	BOOL	8
USINT	USINT	9
SINT	SINT	10
UINT	UINT	2
INT	INT	1
UDINT	UDINT	4
DINT	DINT	3
ULINT	ULINT	27
LINT	LINT	26
REAL	REAL	5
LREAL	LREAL	6
	STRING	12
	WSTRING	21
DATE	DATE	18
	TIME	17
	DATE_AND_TIME	20
TOD	TOD (Time of Day)	19



PLC	zenon	Data type
Long double	Long double	7

#### **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.

# 7.4 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.

## 7.4.1 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.

## 7.4.2 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.

### **A**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	Typ e	Field size	Comment
KANALNAME	Cha r	128	Variable name.
	ı		The length can be limited using the <b>MAX_LAENGE</b> entry in the <b>project.ini</b> file.
KANAL_R	С	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 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	С	128	Identification.  The length can be limited using the MAX_LAENGE entry in the project.ini file.
EINHEIT	С	11	Technical unit
DATENART	С	3	Data type (e.g. bit, byte, word,) corresponds to the data type.
KANALTYP	С	3	Memory area in the PLC (e.g. marker area, data area,) corresponds to the driver object type.
HWKANAL	Nu m	3	Net address
BAUSTEIN	N	3	Datablock address (only for variables from the data area



Identification	Typ e	Field size	Comment
			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)
ОВЈЕКТ	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTTYP and DATENTYP
SIGMIN	Floa t	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



Identification	Typ e	Field size	Comment
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	С	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	С	128	Resources label. Free string for export and display in lists.  The length can be limited using the MAX_LAENGE entry in project.ini.
ADJWVBA	R	1	Non-linear value adaption:  0: Non-linear value adaption is used  1: Non-linear value adaption is not used
ADJZENON	С	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	С	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	Туре	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
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	С	3	Minimum, Maximum
FARBE1	N	10	Color as Windows coding
GRENZTXT1	С	66	Limit value text
A_DELAY1	N	10	Time delay



Identification	Туре	Field size	Comment
INVISIBLE1	R	1	Invisible

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

## 7.5 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	Туре	Offset	Description
MainVersion	UINT	0	Main version number of the driver.



Name from import	Туре	Offset	Description
SubVersion	UINT	1	Sub version number of the driver.
BuildVersion	UINT	29	Build version number of the driver.
RTMajor	UINT	49	zenon main version number
RTMinor	UINT	50	zenon sub version number
RTSp	UINT	51	zenon Service Pack number
RTBuild	UINT	52	zenon build number
LineStateIdle	BOOL	24.0	TRUE, if the modem connection is idle
LineStateOffering	BOOL	24.1	TRUE, if a call is received
LineStateAccepted	BOOL	24.2	The call is accepted
LineStateDialtone	BOOL	24.3	Dialtone recognized
LineStateDialing	BOOL	24.4	Dialing active
LineStateRingBack	BOOL	24.5	While establishing the connection
LineStateBusy	BOOL	24.6	Target station is busy
LineStateSpecialInfo	BOOL	24.7	Special status information received
LineStateConnected	BOOL	24.8	Connection established
LineStateProceeding	BOOL	24.9	Dialing completed
LineStateOnHold	BOOL	24.10	Connection in hold
LineStateConferenced	BOOL	24.11	Connection in conference mode.
LineStateOnHoldPendConf	BOOL	24.12	Connection in hold for conference
LineStateOnHoldPendTransfe r	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 <i>driver stop</i> , the variable has the value <i>TRUE</i> and an <b>OFF</b> bit. After the driver has



Name from import	Туре	Offset	Description
			started, the variable has the value <i>FALSE</i> and no <b>OFF</b> 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
			<ul><li>2: Connection simulated</li></ul>
			Formating:
			<net address="">:<connection status="">;;;</connection></net>
			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.
			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	Туре	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



Name from import	Туре	Offset	Description
			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.
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).



Name from import	Туре	Offset	Description
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	Туре	Offset	Description
MaxWriteTime	UDINT	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	UDINT	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	UDINT	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	UDINT	41	Shortest time in milliseconds (ms) that is required to read a data block.
WriteErrorCount	UDINT	33	Number of writing errors
ReadSucceedCount	UDINT	35	Number of successful reading attempts
MaxCycleTime	UDINT	22	Longest time in milliseconds (ms) required to read all requested data.
MinCycleTime	UDINT	23	Shortest time in milliseconds (ms) required to read all requested data.
WriteCount	UDINT	26	Number of writing attempts
ReadErrorCount	UDINT	34	Number of reading errors
MaxUpdateTimeNor mal	UDINT	56	Time since the last update of the priority group <b>Normal</b> in milliseconds (ms).
MaxUpdateTimeHigh er	UDINT	57	Time since the last update of the priority group <b>Higher</b> in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	58	Time since the last update of the priority group <b>High</b> in milliseconds (ms).
MaxUpdateTimeHigh	UDINT	59	Time since the last update of the priority group



Name from import	Туре	Offset	Description
est			Highest in milliseconds (ms).
PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

### **ERROR MESSAGE**

Name from import	Туре	Offset	Description
ErrorTimeDW	UDINT	2	Time (in seconds since 1.1.1970), when the last error occurred.
ErrorTimeS	STRING	2	Time (in seconds since 1.1.1970), when the last error occurred.
RdErrPrimObj	UDINT	42	Number of the PrimObject, when the last reading error occurred.
RdErrStationsName	STRING	43	Name of the station, when the last reading error occurred.
RdErrBlockCount	UINT	44	Number of blocks to read when the last reading error occurred.
RdErrHwAdresse	DINT	45	Hardware address when the last reading error occurred.
RdErrDatablockNo	UDINT	46	Block number when the last reading error occurred.
RdErrMarkerNo	UDINT	47	Marker number when the last reading error occurred.
RdErrSize	UDINT	48	Block size when the last reading error occurred.
DrvError	USINT	25	Error message as number
DrvErrorMsg	STRING	30	Error message as text
ErrorFile	STRING	15	Name of error log file

# 8 Driver-specific functions

The driver supports the following functions:



The CAN driver supports the reading and writing of data on the CAN bus.

Further functions are not supported by this driver.

### 9 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!

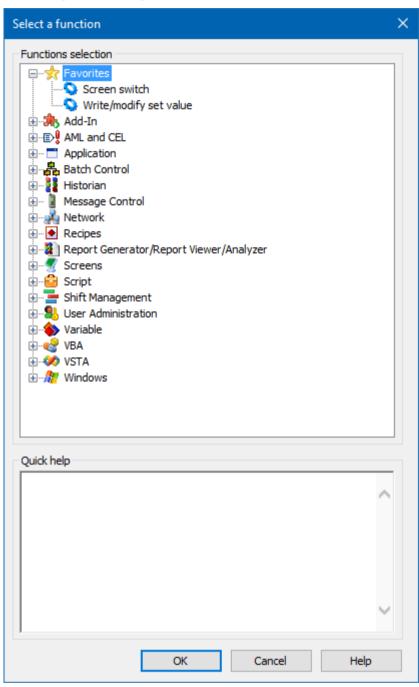
#### **CONFIGURATION OF THE FUNCTION**

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

1. Create a new function in the zenon Editor.



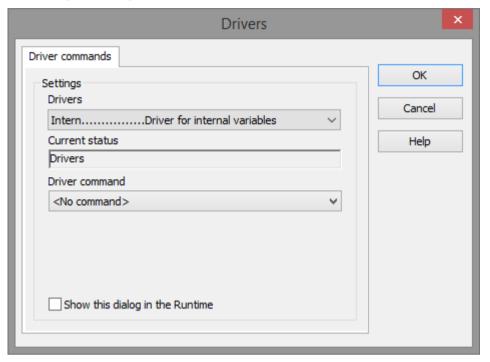
The dialog for selecting a function is opened



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

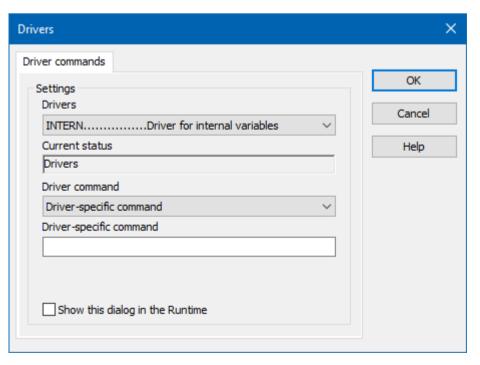


The dialog for configuration is opened



- 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.
	<b>Note:</b> Only available if, for the <b>driver command</b> option, the <i>driver-specific command</i> has been selected.
Show this dialog in the Runtime	Configuration of whether the configuration can be changed in the Runtime:
	<ul> <li>Active: This dialog is opened in the Runtime before executing the function. The configuration can thus still be changed in the Runtime before execution.</li> </ul>
	► Inactive: The Editor configuration is applied in the Runtime when executing the function.
	Default: inactive

### **CLOSE DIALOG**

Options	Description
ОК	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. <b>Note:</b> 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.
	<b>Note:</b> If the driver is in offline mode, all variables that were created for this driver receive the status <i>switched off</i> ( <i>OFF</i> ; Bit <i>20</i> ).
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 $\theta$ and incremented in the set update time by $\theta$ each time up to the maximum value and then start at $\theta$ 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.

# 10 Error analysis

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

# 10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer 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.

### **A**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.

# 10.2 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.</time></drvid></drvdesc></drvexe>	No communication with driver within the given time.
	<time>: Time (in milliseconds)</time>
	<drvdesc>: Driver name</drvdesc>
	<drvexe>: Driver EXE name</drvexe>
	<drvid>: Driver ID in the zenon project</drvid>
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 <i>INVALID</i> 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.

# 10.3 Check list

Questions and hints for fault isolation:

Problem	Diagnostics	Reason
The driver dialog cannot be started.	The driver does not start in the zenon Editor	The necessary software for the Lawicel CAN USB Converter is not installed on the computer. The software is automatically installed on the computer if the USB converter is connected once.
No values can be read or written by the CAN bus.  The bus adapter is connected to a USB interface	The PLC or the CAN bus participant is not connected to the power supply and/or the bus system.	
	<ul> <li>The USB adapter is not correctly connected to the interface (computer's USB port).</li> </ul>	
		The CAN message ID of the message block does not match the expected data.
		► The cable is assigned incorrectly or



Problem	Diagnostics	Reason
		defective.
Certain values cannot be read or written by the PLc.	Has an analysis with the Diagnosis Viewer been carried out to see which errors have occurred?	<ul> <li>Are the variables correctly addressed?</li> <li>Are the correct object types used in the variable?</li> <li>Note: Variables of the driver variable object type do not communicate with the hardware. These only serve for internal driver monitoring.</li> </ul>
Incorrect values are displayed.	Has an analysis with the Diagnosis Viewer been carried out to see which errors have occurred?	<ul> <li>Are the variables correctly addressed?</li> <li>Are the correct data types used?</li> <li>Is the value calculation correct?</li> </ul>

#### **GENERAL TROUBLESHOOTING**

- Is the PLC connected to the power supply?
- Analysis with the help of the Diagnosis Viewer (on page 42):
  - -> Which messages are displayed?
- Can the telegrams be seen with the CAN monitor?
- ▶ Has the USB driver for the USB CAN converter been installed correctly?
- ▶ Has the correct CAN interface been selected?
- ▶ Do the communication parameters (CAN Baud rate) match?
- Is the USB port blocked by another application?
- ▶ Does the offset addressing of the variable match the one in the PLC?

#### SOME VARIABLES REPORT INVALID.

No telegrams have been received from this MessagelD

### VALUES ARE NOT DISPLAYED, NUMERIC VALUES REMAIN EMPTY

Driver is not working. Check the:

- Installation of zenon
- the driver installation
- ▶ The installation of all components
  - -> Pay attention to error messages during the start of the Runtime.



#### VARIABLES ARE DISPLAYED WITH A BLUE DOT

The communication in the network is faulty:

- With a network project:
  Is the network project also running on the server?
- With a stand-alone project or a network project which is also running on the server: Deactivate the property Read from Standby Server only in node Driver connection/Addressing.

#### **VALUES ARE DISPLAYED INCORRECTLY**

Check the information for the calculation in node **Value calculation** of the variable properties.

#### **DRIVER FAILS OCCASIONALLY**

Analysis with the help of the Diagnosis Viewer (on page 42):

-> Which messages are displayed?