## COPADATA

## zenon driver manual

Telco_32

## COPADATA

©2016 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 qualities in the legal sense. Subject to change, technical or otherwise.

## Contents

1. Welcome to COPA-DATA help ..... 4
2. Telco_32 ..... 4
3. TELCO_32-Data sheet ..... 5
4. Driver history ..... 7
5. Requirements ..... 7
6. Configuration ..... 7
6.1 Creating a driver .....  8
6.2 Settings in the driver dialog ..... 10
6.2.1 General ..... 10
6.2.2 Driver dialog Telco 200 settings ..... 13
7. Creating variables ..... 14
7.1 Creating variables in the Editor ..... 15
7.2 Addressing. ..... 18
7.3 Driver objects and datatypes ..... 20
7.3.1 Driver objects ..... 20
7.3.2 Mapping of the data types ..... 21
7.4 Creating variables by importing ..... 23
7.4.1 XML import ..... 24
7.4.2 DBF Import/Export ..... 24
7.5 Driver variables ..... 29
8. Driver-specific functions ..... 36
9. Driver commands ..... 37
10. Error analysis ..... 40
10.1 Analysis tool ..... 40

## 1. Welcome to COPA-DATA help

## GENERAL HELP

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

## PROJECT SUPPORT

You can receive support for any real project you may have from our Support Team, who you can contact via email at support@copadata.com (mailto:support@copadata.com).

## LICENSES AND MODULES

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

## 2. Telco_32

## FOR WHICH PLCS

The driver establishes the connection between zenon and a TELCO200 telecontrol central station.

## CONNECTION

The driver uses the serial interface exclusively for communicating with the central station of the telecontrol system. The central station can connect with external stations to read data or forward commands.

Certain zenon variables can be used to query the status of the central station and the external stations. The different stations are equipped with several modules that perform different tasks (messages, measured values, counters, sensors, commands). The numbers of the modules are saved in the file TELCO200.INI in the subfolder custom\drivers of the Runtime directory.

## 3. TELCO_32-Data sheet

| General: |  |
| :--- | :--- |
| Driver file name | TELCO_32.exe |
| Driver name | Telco200 driver |
| PLC types | Telco200Z, Telco210Z |
| PLC manufacturer | Telco; Vollmer; |


| Driver supports: |  |
| :--- | :--- |
| Protocol | unknown; |
| Addressing: Address-based | X |
| Addressing: Name-based | -- |
| Spontaneous <br> communication | -- |
| Polling communication | X |
| Online browsing | -- |
| Offline browsing | -- |
| Real-time capable | -- |
| Blockwrite | -- |
| Modem capable | -- |
| Serial logging | -- |
| RDA numerical | X |
| RDA String | -- |


| Requirements: |  |
| :--- | :--- |
| Hardware PC | RS232 serial interface |
| Software PC | -- |
| Hardware PLC | Telco LAN-interface |
| Software PLC | -- |
| Requires v-dll | -- |

## Platforms:

| Operating systems | Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server <br> 2012R2; |
| :--- | :--- |
| CE platforms | $-;$ |

## 4. Driver history

| Date | Driver version | Change |
| :--- | :--- | :--- |
| 07.07 .08 | 100 | 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: $\mathbf{7 . 1 0 . 0 . 4 2 2 8}$ means: The driver is for version $\mathbf{7 . 1 0}$ service pack $\mathbf{0}$, and has the build number 4228.

Expansions or error rectifications will be incorporated into a build in the future and are then available form 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. Requirements

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

## 6. Configuration

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

## Q Information <br> Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) of the online manual.

### 6.1 Creating a driver

In order to create a new driver:

1. Right-click on Driver in the Project Manage and select Driver new in the context menu.
2. In the following dialog the control system offers a list of all available drivers.

Definition of driver...

Available drivers


Driver name
Codesys Arti NG driver

Driver information
Description:
Codesys Soft PLC driver "New Generation" for the Codesys 3S-ARTI (Asynchron Runtime Interface) interface. The driver supports direct variable import from the Codesys development environment and replaces the Codesys Arti Soft PLC driver.

Supported PLC types:
Codesys Soft PLCs, Moeller XControl PLCs XC200 and XC600, and Elau PacDrive Controller MAX 4, C200, C400, C600, P600.

Supported connection types:
Ethernet; Local
3. Select the desired driver and give it a name:

- The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.
- The driver name is part of the file name. Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (_).
- Attention: This name cannot be changed later on.

4. Confirm the dialog with OK. In the following dialog the single configurations of the drivers are defined.

Only the respective required drivers need to be loaded for a project. Later loading of an additional driver is possible without problems.

```
    8 Information
    For new projects and for existing projects which are converted to version 6.21 or higher,
    the following drivers are created automatically:
* Internal
MathDr32
| SysDrv.
```


### 6.2 Settings in the driver dialog

You can change the following settings of the driver:

### 6.2.1 General

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.


| Parameters | Description |
| :---: | :---: |
| Mode | Allows to switch between hardware mode and simulation mode <br> Hardware: <br> A connection to the control is established. <br> Simulation static <br> 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. <br> Simulation - counting <br> No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically. <br> Simulation - programmed <br> N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm). |
| Keep update list in the memory | Variables which were requested once are still requested from the control even if they are currently not needed. <br> This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control. |
| Output can be written | Active: Outputs can be written. <br> Inactive: Writing of outputs is prevented. <br> 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. <br> Fundamental requirement: The variable must have a valid value and time stamp. |


|  | The variable image is saved in mode hardware if: <br> one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active <br> The variable image is always saved if: <br> - the variable is of the object type Driver variable <br> - the driver runs in simulation mode. (not programmed simulation) <br> The following states are not restored at the start of the Runtime: <br> - SELECT(8) <br> - WR-ACK(40) <br> - WR-SUC(41) <br> The mode Simulation - programmed at the driver start is not a criterion in order to restore the remanent variable image. |
| :---: | :---: |
| Stop on Standby Server | Setting for redundancy at drivers which allow only on communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade. <br> Attention: If this option is active, the gapless archiving is no longer guaranteed. <br> Active: Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status switched off (statusverarbeitung.chm::/24150.htm) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian. <br> Note: Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter. |
| Global Update time | Active: The set Global update time in ms is used for all variables in the project. The priority set at the variables is not used. <br> Inactive: The set priorities are used for the individual variables. |
| Priority | The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time. <br> The allocation to the variables takes place separately in the settings of the variable properties. <br> The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities. |


|  | Thus the communication load is distributed better. <br> Attention: Priority classes are not supported by each driver For <br> example, drivers that communicate spontaneously do not <br> support it. |
| :--- | :--- |

## CLOSE DIALOG

| Parameters | Description |
| :--- | :--- |
| OK | Applies all changes in all tabs and closes the dialog. |
| Cancel | Discards all changes in all tabs and closes the dialog. |
| Help | Opens online help. |

## UPDATE TIME FOR CYCLICAL DRIVERS

The following applies for cyclical drivers:
For Set value, Advising of variables and Requests, a read cycle is immediately triggered for all drivers regardless of the set update time. This ensures that the value is immediately available for visualization after writing. Update times can therefore be shorter than pre-set for cyclical drivers.

### 6.2.2 Driver dialog Telco 200 settings



## CYCLE TIMES

| Parameters | Description |
| :--- | :--- |
| Basic cycle | (Internal) basic cycle for request and setclock cycles |
| Scanning cycle | Cycle at which the stations will be scanned |
| Set clock | Cycle at which the clock of the central station will be synchronized |

## NUMBER OF ATTEMPTS

| Parameters | Description |
| :--- | :--- |
| A command | Number of retries if the sending of an A command to the central station fails |
| M command | Number of retries if the sending of an $M$ command to the central station fails |

STACK

| Parameters | Description |
| :--- | :--- |
| Size | Size of the command stack. The command stack contains the upcoming <br> commands that will be executed next. |

## MODULES

| Parameters | Description |
| :--- | :--- |
| Modules | Number of the modules in the different module types. They are read from the <br> file TELCO200.INI (see below). Here, they are only displayed. |

## CFG FILE

| Parameters | Description |
| :--- | :--- |
| CFG file | Name of the file that contains the process image of the driver (always <br> TELCO200.DAT, only displayed here) |

The settings are saved in the project database (for earlier versions in zenon.ini)

## 7. Creating variables

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

### 7.1 Creating variables in the Editor

Variables can be created:

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


## VARIABLE DIALOG

To create a new variable, regardless of which type:

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

2. The dialog for configuring variables is opened
3. configure the variable
4. The settings that are possible depends on the type of variables


| Property | Description |
| :--- | :--- |
| Name | Distinct name of the variable. If a variable with the same name already <br> exists in the project, no additional variable can be created with this name. <br> Maximum length: 128 character <br> Attention: The characters \# and @ are not permitted in variable names. If <br> non-permitted characters are used, creation of variables cannot be <br> completed and the Finish button remains inactive. <br> Note: For some drivers, the addressing is possible over the property <br> Symbolic address, as well. |
| Drivers | Select the desired driver from the drop-down list. <br> Note: If no driver has been opened in the project, the driver for internal <br> variables (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) is <br> automatically loaded. |
| Driver object type |  |
| (cti.chm::/28685.htm) | Select the appropriate driver object type from the drop-down list. |
| Data type | Select the desired data type. Click on the ... button to open the selection <br> dialog. |
| Array settings | Expanded settings for array variables. You can find details in the Arrays <br> chapter. |
| Addressing options | Expanded settings for arrays and structure variables. You can find details <br> in the respective section. |
| Automatic element <br> activation | Expanded settings for arrays and structure variables. You can find details <br> 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.

## 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 | Freely definable name. |
| Attention: For every zenon project the name must be unambiguous. |  |$|$| Freely assignable identification, e.g. for resources label, comment ... |  |
| :--- | :--- |
| Identification | Bus address or net address of the variable. <br> This address refers to the bus address in the connection configuration of the <br> driver. This defines the PLC, on which the variable resides. |
| Net address | For variables of object type Extended data block, enter the datablock number <br> here. <br> Adjustable from 0 to $4294967295 . ~ Y o u ~ c a n ~ t a k e ~ t h e ~ e x a c t ~ m a x i m u m ~ a r e a ~ f o r ~$ |
| data blocks from the manual of the PLC. |  |

Compared to earlier versions, the names of the address elements have changed. Instead of the terms "Station", "Module" and "Channel" previously used, the standard terms "Net address", "Data block" and "Offset" are used in the current version.

| Old term | New term |
| :--- | :--- |
| Station | Net address |
| Module | Data block |
| Channel | Offset |

## CONFIGURATION FILE

The file telco200.ini in the subdirectory custom\drivers of the project directory contains the definition of the module numbers of the stations (in earlier versions zenon.ini).

The file has the following structure: ( $x$ is a consecutive index that allows you to define several discontiguous number ranges for every module type)
[TELCO200]

MOD_MELDUNG_VON_x=
MOD_MELDUNG_BIS_x=

MOD_MESSWERT8_VON_x=
MOD_MESSWERT8_BIS_x=

MOD_MESSWERT12_VON_x=
MOD_MESSWERT12_BIS_x=

Sector ID, always TELCO200

The valid module numbers for the messages
$x=1,2,3, \ldots$ (numbered consecutively, starting with 1) (MOD_MELDUNG_VON_x<=MOD_MELDUNG_BIS_x)

The valid module numbers for the 8 -bit measured values
$x=1,2,3, \ldots$ (numbered in pairs, starting with 1 )
(MOD_MESSWERT8_VON_x<=MOD_MESSWERT8_BIS_x)

The valid module numbers for the 12 bit measured values $x=1,2,3, \ldots$. (numbered in pairs, starting with 1 )
(MOD_MESSWERT12_VON_x<=MOD_MESSWERT12_BIS_x)

| MOD_ZAEHLWERT_VON_x= | The valid module numbers for the counted values |
| :---: | :---: |
| MOD_ZAEHLWERT_BIS_x= | $x=1,2,3, \ldots$ (numbered in pairs, starting with 1 ) <br> (MOD_ZAEHLWERT _VON_x<=MOD_ZAEHLWERT _BIS_x) |
| MOD_BEFEHL_VON_x= | The valid module numbers for the commands |
| MOD_BEFEHL_BIS_x= | $x=1,2,3, \ldots$ (numbered in pairs, starting with 1$)$ <br> (MOD_BEFEHL _VON_x<=MOD_BEFEHL_BIS_x) |
| MOD_MESSGEBER_VON_x= | The valid module numbers for the sensors |
| MOD_MESSGEBER_BIS_x= | $x=1,2,3, \ldots$ (numbered in pairs, starting with 1$)$ <br> (MOD_BEFEHL _VON_x<=MOD_BEFEHL_BIS_x) |
| STAPELZEIGEN=0 | If set to 1, the command "Display number of commands |

### 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 <br> types | Channel <br> type | Supported <br> datatypes | Read |
| :--- | :--- | :--- | :--- | :--- | :--- |
| (DataType) |  |  |  | Write | Comment |
| :--- |
| Central station <br> status |
| External station <br> status |
| (80 |
|  |

### 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.

## CENTRAL STATION STATUS

For every driver, you can only create one variable of this type. The values have the following meaning:

| Direction | Value | Meaning |
| :--- | :--- | :--- |
| Read | 0 | "Computer system TELCO 4000 and data recording system in operation" |
|  | 1 | "System error computer system TELCO 4000" |
|  | 2 | "Dialing to external station" |



## EXTERNAL STATION STATUS

For every external station, you can only create two variables of this type: one UINT variable and one STRING variable each. The variable with the data type STRING stores the time of the last connection with the external station.

The variable with the data type UINT is used to transmit status information about the external station:

| Direction | Value | Meaning |
| :--- | :--- | :--- |
| Read | 0 | "Computer system TELCO 4000 and data recording system in operation" |
| 2 | "Dialling to external station" |  |
|  | 4 | "Dialing to external station failed" |
|  | "Connection to external station active" |  |
|  | 16 | "Terminating connection to external station" |
|  | 32 | "Transmitting command or set value to external station" |
|  | 64 | "Updating measurands" |
|  | 128 | "Measurands update canceled" |

zenon

|  | 256 | "Error in zenon command" (e.g. "E" command without <br> establishing a connection first; "A" command when there is already a <br> connection to another station; ...) |
| :--- | :--- | :--- |
|  | 512 | Stack overflow <br> Write |
|  | 1 | Command "A" - Establish a connection to the external station and fetch data |
|  | 3 | Delete the status channel of the station (set to 0). |

## SENSOR

After an M command, the driver reads data from a sensor. This data is then sent, like all other data, to the zenon RT module via the client-server interface. The data is then processed by the Historian in the RT and sorted into the archives. The sensor variables must therefore be marked as "post-sorted" in the hard-disk data storage properties group.

The M command is triggered by writing the value 1 to the according sensor.
Writing the configured maximum value of the variable will display a dialog with the current status of the sensor.

Data type: The property Data type is the internal numerical name of the data type. It is also used for the extended DBF import/export of the variables.

### 7.4 Creating variables by importing

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

## 8 Information

You can find details on the import and export of variables in the Import-Export (main.chm::/13028.htm) manual in the Variables (main.chm::/13045.htm) section.

### 7.4.1 XML import

For the import/export of variables the following is true:

- The import/export must not be started from the global project.
- The start takes place via:
- Context menu of variables or data typ in the project tree
- or context menu of a variable or a data type
- or symbol in the symbol bar variables


## $\triangle$ Attention

When importing/overwriting an existing data type, all variables based on the existing data type are changed.

Example:
There is a data type XYZ derived from the type INTwith variables based on this data type. The XML file to be imported also contains a data type with the name XYZ but derived from type STRING. If this data type is imported, the existing data type is overwritten and the type of all variables based on it is adjusted. I.e. the variables are now no longer INT variables, but STRING variables.

### 7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.

## 8 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 import assistant

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

## 8 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 export assistant

## © Attention

DBF files:

- must correspond to the 8.3 DOS format for filenames (8 alphanumeric characters for name, 3 character suffix, no spaces)
b must not have dots (.) in the path name.
e.g. the path C: \users $\backslash$ John. Smith $\backslash$ test. $\mathrm{db} f$ 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.

## Q 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:
v 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. <br>  |
|  | The length can be limited using the MAX_LAENGE entry in <br> project.ini. |  |

KANAL_R C 128
KANAL_D Log 1
TAGNR C 128

| EINHEIT | C | 11 |
| :--- | :--- | :--- |
| DATENART | C | 3 |
| KANALTYP | C | 3 |


| HWKANAL | Num | 3 |
| :--- | :--- | :--- |
| BAUSTEIN | N | 3 |


| ADRESSE | N | 5 |
| :--- | :--- | :--- |
| BITADR | N | 2 |

ARRAYSIZE N 16

| LES_SCHR | L | 1 | Write-Read-Authorization <br> 0 : Not allowed to set value. <br> 1: Allowed to set value. |
| :---: | :---: | :---: | :---: |
| MIT_ZEIT | L | 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 |
| 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) <br> 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 | L | 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 | L | 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. <br> Free string for export and display in lists. <br> The length can be limited using the MAX_LAENGE entry in project.ini. |
| ADJWVBA | L | 1 | Non-linear value adaption: <br> 0 : Non-linear value adaption is used <br> 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. |
| $\triangle$ At |  |  |  |

## LIMIT VALUE DEFINITION

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

| Identification | Type | Field size | Comment |
| :---: | :---: | :---: | :---: |
| AKTIV1 | L | 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) <br> (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 | L | 1 | Set blink attribute |
| BTB1 | L | 1 | Logging in CEL |
| ALARM1 | L | 1 | Alarm |
| DRUCKEN1 | L | 1 | Printer output (for CEL or Alarm) |
| QUITTIER1 | L | 1 | Must be acknowledged |
| LOESCHE1 | L | 1 | Must be deleted |
| VARIABLE1 | L | 1 | Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTX). |
| FUNC1 | L | 1 | Functions linking |
| ASK_FUNC1 | L | 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 | L | 1 | Invisible |

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

### 7.5 Driver variables

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

- Information
- Configuration
- Statistics and
- Error message

The definitions of the variables implemented in the driver kit are available in the import file drvvar.dbf (on the installation medium in the \Predefined $\backslash$ Variables folder) and can be imported from there.

Note: Variable names must be unique in zenon. If driver variables are to be imported from drvvar.dbf again, the variables that were imported beforehand must be renamed.

## Information

Not every driver supports all driver variants.
For example:

- Variables for modem information are only supported by modem-compatible drivers
- Driver variables for the polling cycle only for pure polling drivers
- Connection-related information such as ErrorMSG only for drivers that only edit one connection at a a time


## INFORMATION

| Name from import | Type | Offset | Description |
| :--- | :--- | :--- | :--- |
| MainVersion | UINT | 0 | Main version number of the driver. |
| SubVersion | UINT | 1 | Sub version number of the driver. |
| BuildVersion | UINT | 29 | Build version number of the driver. |
| RTMajor | UINT | 49 | zenon main version number |
| RTMinor | UINT | 51 | zenon Service Pack number |
| RTSp | UINT | 52 | zenon build number |
| RTBuild | BOOL | 24.0 | TRUE, if the modem connection is idle |
| LineStateldle | BOOL | 24.1 | TRUE, if a call is received |
| LineStateOffering | BOOL | 24.2 | The call is accepted |
| LineStateAccepted | BOOL | 24.3 | Dialtone recognized |
| LineStateDialtone | BOOL | 24.4 | Dialing active |
| LineStateDialing | 24.5 | While establishing the connection |  |
| LineStateBusy | 24.6 | Target station is busy |  |
|  |  |  |  |


| LineStateSpeciallnfo | 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. |  |  |  |

## CONFIGURATION

| Name from import | Type | Offset | Description |
| :--- | :--- | :--- | :--- |
| ReconnectInRead | BOOL | 27 | If TRUE, the modem is automatically <br> reconnected for reading |
| ApplyCom | BOOL | 36 | Apply changes in the settings of the serial <br> interface. Writing to this variable <br> immediately results in the method <br> SrvDrvVarApplyCom being called (which <br> currently has no further function). |
| ApplyModem | BOOL | 37 | Apply changes in the settings of the <br> modem. Writing this variable immediately <br> calls the method SrvDrvVarApplyModem. <br> This closes the current connection and <br> opens a new one according to the settings <br> 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 <br> 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 <br> connection in milliseconds (ms). |
| :--- | :--- | :--- | :--- |
| RingCountSet | UDINT | 20 | Number of ringing tones before a call is <br> accepted |
| ReCallIdleTime | UINT | 53 | Waiting time between calls in seconds (s). |
| ConnectTimeout | UINT | 54 | Time in seconds (s) to establish a <br> connection. |

## STATISTICS

| Name from import | Type | Offset | Description |
| :--- | :--- | :--- | :--- |
| MaxWriteTime | UDINT | 31 | The longest time in milliseconds (ms) that is <br> required for writing. |
| MinWriteTime | UDINT | 32 | The shortest time in milliseconds (ms) that is <br> required for writing. |
| MaxBIkReadTime | UDINT | 40 | Longest time in milliseconds (ms) that is required <br> to read a data block. |
| MinBIkReadTime | UDINT | 41 | Shortest time in milliseconds (ms) that is required <br> 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 <br> all requested data. |
| :--- | :--- | :--- | :--- |
| MinCycleTime | UDINT | 23 | Shortest time in milliseconds (ms) required to read <br> 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 <br> Normal in milliseconds (ms). |
| MaxUpdateTimeHigher | UDINT | 57 | Time since the last update of the priority group <br> Higher in milliseconds (ms). |
| MaxUpdateTimeHigh | UDINT | 58 | Time since the last update of the priority group <br> High in milliseconds (ms). |
| MaxUpdateTimeHighest | UDINT | 59 | Time since the last update of the priority group <br> Highest in milliseconds (ms). |
| PokeFinish | BOOL | 55 | Goes to 1 for a query, if all current pokes were <br> executed |

## ERROR MESSAGE

| Name from import | Type | Offset | Description |
| :--- | :--- | :--- | :--- |
| ErrorTimeDW | UDINT | 2 | Time (in seconds since 1.1.1970), when the last error <br> occurred. |
| ErrorTimeS | STRING | 2 | Time (in seconds since 1.1.1970), when the last error <br> occurred. |
| RdErrPrimObj | UDINT | 42 | Number of the PrimObject, when the last reading error <br> 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 <br> 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:

## DRIVER SUPPORTS

## ERROR FILE

Yes

## EXTENDED ERROR FILE

No

SERIAL LOGGING

Yes. See general driver documentation

## BLOCKWRITE

No

## REDUNDANCY

Can the driver be operated in redundant mode?

RDA

Yes, but only for variables of the type "Sensor"

REAL TIME STAMPING

Yes

## NUMBER OF PLCS

One central station with any number of external stations

## WINCE

No

## 9. Driver commands

This chapter describes standard functions that are valid for most zenon drivers. Not all functions described here are available for every driver. For example, a driver that does not, according to the data sheet, support a modem connection also does not have any modem functions.

Driver commands are used to influence drivers using zenon; start and stop for example.
The engineering is implemented with the help of function Driver commands. To do this:

- create a new function
- select Variables -> Driver commands
- The dialog for configuration is opened


| Parameter | Description |
| :---: | :---: |
| Drivers | Drop-down list with all drivers which are loaded in the project. |
| Current status | Fixed entry which has no function in the current version. |
| Driver command | Drop-down list for the selection of the command. |
| Start driver (online mode) | Driver is reinitialized and started. |
| Stop driver (offline mode) | Driver is stopped. No new data is accepted. <br> 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. <br> 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. <br> For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed. |
| , Driver-specific command | Enter driver-specific commands. Opens input field in order to enter a command. |
| Driver - activate set setpoint value | Write set value to a driver is allowed. |
| 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) |
| Show this dialog in the Runtime | The dialog is shown in Runtime so that changes can be made. |

## DRIVER COMMANDS IN THE NETWORK

If the computer, on which the driver command function is executed, is part of the zenon network, additional actions are carried out. A special network command is sent from the computer to the project server, which then executes the desired action on its driver. In addition, the Server sends the same driver command to the project standby. The standby also carries out the action on its driver.

This makes sure that Server and Standby are synchronized. This only works if the Server and the Standby both have a working and independent connection to the hardware.

## 10. Error analysis

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

### 10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (main.chm::/12464.htm) program that was also installed with zenon. You can find it under Start/All programs/zenon/Tools 7.50 -> Diagviewer.
zenon driver log all errors in the LOG files. The default folder for the LOG files is subfolder LOG in directory ProgramData, example:
\%ProgramData\% \COPA-DATA \LOG. LOG files are text files with a special structure.
Attention: With the default settings, a driver only logs error information. With the Diagnosis Viewer you can enhance the diagnosis level for most of the drivers to "Debug" and "Deep Debug". With this the driver also logs all other important tasks and events.

In the Diagnosis Viewer you can also:

- Follow 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.
