zenon driver manual

3S_Arti

v.7.00

COPADATA
do it your way

Contents

1. Welcome to COPA-DATA help

GENERAL HELP

If you miss any information in this help chapter or have any suggestions for additions, please feel free to contact us via e-mail: documentation@copadata.com (mailto:documentation@copadata.com).

PROJECT SUPPORT

If you have concrete questions relating to your project, please feel free to contact the support team via e-mail: support@copadata.com (mailto:support@copadata.com)

LICENSES AND MODULES

If you realize that you need additional licenses or modules, please feel free to contact the sales team via e-mail: sales@copadata.com (mailto:sales@copadata.com)

2. 3S_Arti

With the defined communication parameters the driver establishes a connection to the Runtime system of the SoftPLC.

CONVERTING FROM VERSION 5 TO VERSION 6

After converting the driver configuration should be opened again and be confirmed with "OK".

Then the directory "drivers" in \ED\FILES\zenon\custom is created.

Do not browse the PLC and add the variables, before the old variable file has been copied as described below.

LIMITATIONS OF USE UNDER VERSION 5

In version 5 the driver writes an allocation table with the name <projectname>_codesys.cmp; this file is in the project directory.

The driver cannot be started several times, because only one such file can exist.

EXTENSIONS OF THE DRIVER IN VERSION 6.01 SERVICE PACK 2

With version 6.01 service pack 2 the variable name is preceded by the station address acting as an index. This means that when the variables are imported again from the symbol table, these variables are created twice. Thus it can happen that there are two variables with the same name in one project, distinguished only by the station index.

This extension of the variable name is however no functional restriction of the existing variable.

3. 3S_ARTI - Data sheet

General:	
Driver file name	3S_ARTI.exe
Driver description	Codesys Arti driver
PLC types	Codesys Soft PLCs, Moeller XControl PLCs XC200 and XC600, and Elau PacDrive Controller MAX 4, C200, C400, C600, P600.
PLC manufacturer	3S; Elau; Moeller;

Driver supports:	
Protocol	3S-Arti;
Addressing: address based	-
Addressing: name based	x
Spontaneous communication	-
Polling communication	x
Online browsing	x
Offline browsing	x
Real-time capable	-
Blockwrite	-
Modem capable	-
Serial logging	-
RDA numerical	-
RDA String	-

Prerequisites:	
Hardware PC	Standard network card
Software PC	Codesys software version 2.12 or higher incl. ARTI interface; Windows CE: SymArticlient.dll and Articlient.dll necessary, has to be ordered from 3S.
Hardware PLC	-
Software PLC	Codesys Software version 2.12 or higher incl. ARTI interface
Requires v-dll	-

Platforms:	
Operating systems	Windows CE 5.0, CE 6.0; Windows XP, Vista, 7, Server 2003, Server 2008/R2;
CE platforms	x86; ARM; Pocket-PC;

4. Driver history

Date	Driver version	Comment
08.02.2002		Creating the documentation
05.05.2003		Text file created on browsing the variable list
09.08.2004		Name of the allocation table file changed. Driver now can be started several times. Since driver version 6.1.1.1600
03.03.2005		Added information about the name change of the variables in 6.01 SP2
14.08.2008	2500	

5. Requirements

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

5.1 PC

Copy the driver file 3S_ARTI.EXE to the current zenon directory (unless it is already there).

The driver needs the files ArtiClient.dll and SymArtiClient.dll. These DLLs should be installed along with the CoDeSys software (not provided by COPA-DATA - must be purchased separately).

CE

Under CE the driver 3S_ARTI.DLL is automatically copied to the CE device by the Editor via Remote Transport. The DLLs ArtiClient.dll and SymArtiClient.dll are also needed on the CE device. But there are own DLLs for Windows CE! At the moment there are DLLs for X86 and StrongArm processors. Also these DLLs should be obtained from 3S. On the CE device these files have to be copied to the Runtime directory. (This is also where the files zenon6.ini and zenonrce.exe are stored).

Adding the variables from the PLC in the driver configuration creates a file in the project directory on the PC: 3S_Arti_drivername.dat. This file has to be copied to the project directory on the CE device. Otherwise the communication is displayed as disturbed.

Under Windows CE it is not possible to use several drivers of the same type.

6. Configuration

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

> **Info**
>
> *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:

▸ Right-click on `Driver` in the Project Manage and select `Driver new` in the context menu.

▶ In the following dialog the control system offers a list of all available drivers.

▶ 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, every time 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.

▶ Confirm the dialog with ok. In the following dialog the single configurations of the drivers are defined.

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

> 💡 **Info**
>
> *For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:*
>
> ▸ Internal
>
> ▸ MathDr32
>
> ▸ SysDrv.

▸

6.2 Settings in the driver dialog

You can change the following settings of the driver:

6.2.1 General

Parameters	Description
Mode	Allows to switch between hardware mode and simulation mode
	▶ Hardware:
	A connection to the control is established.
	▶ Simulation static
	No communication between to the control is established, the values are simulated by the driver. In this modus the values remain constant or the variables keep the values which were set by zenon Logic. Each variable has its own memory area, e.g. two variables of the type marker with offset 79 can have different values in the Runtime and do not influence each other. Exception: The simulator driver.
	▶ Simulation - counting
	No communication between to the control is established, the values are simulated by the driver. In this modus the driver increments the values within a value range automatically.
	▶ Simulation - programmed
	N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).
Keep update list in the memory	Variables which were requested once are still requested from the control even if they are currently not needed. This has the advantage that e.g. multiple screen switches after the screen was opened for the first time are executed faster because the variables need not be requested again. The disadvantage is a higher load for the communication to the control.
Output can be written	Active: Outputs can be written. Inactive: Writing of outputs is prevented. Note: Not available for every driver.

`Variable image remanent`	This option saves and restores the current value, time stamp and the states of a data point. Fundamental requirement: The variable must have a valid value and time stamp. The variable image is saved in mode hardware if: ▸ one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active The variable image is always saved if: ▸ the variable is of the object type `Driver variable` ▸ the driver runs in simulation mode. (not programmed simulation) The following states are not restored at the start of the Runtime: ▸ SELECT(8) ▸ WR-ACK(40) ▸ WR-SUC(41) The mode **`Simulation - programmed`** at the driver start is not a criterion in order to restore the remanent variable image.
`Stop at the Standby Server`	Setting for redundancy at drivers which allow only on communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade. **Attention:** If this option is active, the gapless archiving is no longer guaranteed. `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.
`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. `Inactive`: The set priorities are used for the individual variables.
`Priority`	Here you set the polling times for the individual priorities. All variables with the according priority are polled in the set time. The allocation is taken

	place for each variable separately in the settings of the variable properties. The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities. Thus the communication load is distributed better.

UPDATE TIME FOR CYCLICAL DRIVER

The following applies for cyclical drivers:

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

6.2.2   Driver dialog CoDeSys

RUNTIME

Setting the communication parameters

Local: Runtime system runs on the same computer

Remote: Runtime system runs on another computer; select the computer with the button « … »

Port: Default 1200

Protocol: TCP/IP Level 2 Route or Level 4

Buffer size: Default 5000

VARIABLE DEFINITION

Selection from which Runtime system the symbol table should be read

6.3   Speeding up the starting of Runtime

If there are a large amount of variables on the PLC, this can delay the start of zenon Runtime or the initial switching of a screen in Runtime. The delay is caused by the following behavior:

▸   The symbol table of the PLC is read in to the file `SymArtiClient.dll` when the connection is first made

▸   This symbol table can be very large, depending on the variables on the PLC

▸   The size of the file determines the duration of the delay
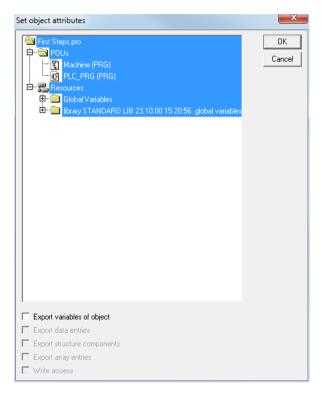
To reduce the connection time, the symbol table can be reduced in the `EPAS/CodeSys` software. To do this, only the variables that are to communicate with zenon are integrated.
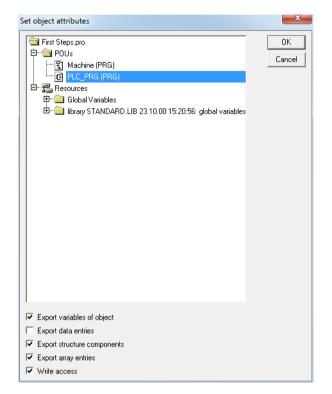
REDUCING THE SYMBOL TABLE

To reduce the symbol table:

1. Start the **EPAS/CodeSys** software

2. Deactivate all options for the symbol table
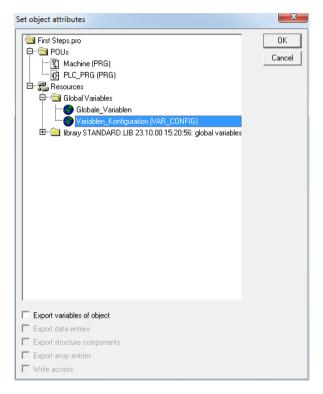
3. Select the modules of the PLC that are to have their variables integrated into the symbol table

4. You can also select individual variables, for example from the group of `global variables`.

The possibility to select `global variables` can differ with different software versions.

5. If all variables required for communication with zenon are selected, the configuration can be confirmed as usual.

Since there are now fewer variables integrated in the symbol table, this is significantly smaller once it has been transferred to the PLC again. It therefore takes considerably less time to load this file to the `SymArtiClient.dll` file and initial communication between Runtime and the PLC is quicker.

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 exists in the project, no additional variable can be created with this name. **Attention:** The **#** character is not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the **Finish** button remains inactive.
Drivers	Select the desired driver from the drop-down list. **Note:** If no driver has been opened in the project, the driver for internal variables (Intern.exe (Main.chm::/Intern.chm::/Intern.htm)) is automatically loaded.
Driver object type (cti.chm::/28685.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 dialog.
Array settings	Expanded settings for array variables. You can find details in the 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.

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

CODESYS ADDRESS RANGE

With the button "Update list" the list is filled with the contents of the symbol table of the selected Runtime system. For this the SoftPLC has to be in "Run" mode.

ZENON ADDRESS RANGE:

With the button "Add" entries from the symbol table can be transferred to the list. The entries in the list are provided with an index, which is used for the addressing of the zenon variables. The "Delete" button deletes entries from the list. On closing the window with "OK" the list is saved to a file. This file is in the project directory and has the name: 3S_Arti_Treiberbezeichnung.dat.

TEXT FILE WITH LOCAL ADDRESS LIST

Since driver version 5.2.0.4 on pressing the button "Add" a text file is automatically created. This file has the name "3S_Arti_drivername.txt" and contains the following structure:

Name   Type   RefId   Size   Address

.Testvar4   BOOL   4   1   13

The columns "RedId" and "Size" are only used inside CoDeSys and are not used directly by zenon, but with "Name", "Type" and "Address" a DBF import file can be created. This facilitates the creation of the zenon variables.

Allocation to DBF columns:

Text file	DBF file	Comment
Name	KANALNAME	
Type	DATATYPE, CHANNELTYPE, OBJECT	All three columns have to be defined according to the following table (Chapter 4.1).
Address	ADRESSE	

ADDRESSING OF INDIVIDUAL BITS, BINARY VARIABLES

The 3S_Arti driver uses an assignment file in which the information for communication with the PLC is saved. The variable offset is used in zenon to read off information from the assignment file. To address individual bits in an INT:

1. An INT with the offset 4 is imported from the PLC and

2. then 16 binary variables with the offset 4 and the bit numbers 0 to 15 are created

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	Channel type	Read / Write	Supported data types	Comment
PLC marker	8	R / W	LREAL, BOOL, REAL, DINT, UDINT, INT, UINT, STRING, USINT, SINT	

OBJECTS FOR PROCESS VARIABLES IN ZENON

Object	Channel type	Data type	Name
2	8	8	Marker - (M)BOOL
3	8	9	Marker - (MB)BYTE/CHAR
4	8	2 or 1	Marker - (MW)WORD/INT
5	8	4 or 3	Marker - (MD)DWORD/DINT
6	8	5	Marker – (DF)FLOAT
7	8	6	Marker – (DD)DOUBLE
8	8	12	Marker – (DS)STRING

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	8
	USINT	9
	SINT	10
	UINT	2
	INT	1

	UDINT	4
	DINT	3
	ULINT	27
	LINT	26
	REAL	5
	LREAL	6
	STRING	12
	WSTRING	21
	DATE	18
	TIME	17
	DATE_AND_TIME	20
	TOD (Time of Day)	19

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.

7.4.1 XML import of variables from another zenon project

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

> ⚠️ **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 `INT` with 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.

IMPORT DBF FILE

To start the import:

1. right-click on the variable list

2. in the drop-down menu 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.

> 💡 **Info**
>
> *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 menu 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)
>
> ▸ 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.

> 💡 **Info**
>
> *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) when exporting.
>
> DBF files must:
>
> ▸ correspond to the 8.3 DOS format for filenames (8 alphanumeric characters for name, 3 character suffix, no spaces)
>
> ▸ Be stored close to the root directory (Root)

DESIGN

Description	Type	Field size	Comment
KANALNAME	Char	128	Variable name. The length can be limited using the MAX_LAENGE entry in **project.ini** .
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "KANALNAME" (field/column must be entered manually). The length can be limited using the MAX_LAENGE entry in **project.ini** .
KANAL_D	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
TAGNR	C	128	Identification. The length can be limited using the MAX_LAENGE entry in **project.ini** .
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Bus address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset

BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipe Group Manager
LES_SCHR	R	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
MIT_ZEIT	R	1	time stamp in zenon (only if supported by the driver)
OBJEKT	N	2	Driver-specific ID number of the primitive object comprises KANALTYP and DATENART
SIGMIN	Float	16	Non-linearized signal - minimum (signal resolution)
SIGMAX	F	16	Non-linearized signal - maximum (signal resolution)
ANZMIN	F	16	Technical value - minimum (measuring range)
ANZMAX	F	16	Technical value - maximum (measuring range)
ANZKOMMA	N	1	Number of decimal places for the display of the values (measuring range)
UPDATERATE	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
MEMTIEFE	N	7	Only for compatibility reasons
HDRATE	F	19	HD update rate for historical values (in sec, one decimal possible)
HDTIEFE	N	7	HD entry depth for historical values (number)
NACHSORT	R	1	HD data as postsorted values
DRRATE	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
HYST_PLUS	F	16	Positive hysteresis, from measuring range
HYST_MINUS	F	16	Negative hysteresis, from measuring range
PRIOR	N	16	Priority of the variable
REAMATRIZE	C	32	Allocated reaction matrix

ERSATZWERT	F	16	Substitute value, from measuring range
SOLLMIN	F	16	Minimum for set value actions, from measuring range
SOLLMAX	F	16	Maximum for set value actions, from measuring range
VOMSTANDBY	R	1	Get value from standby server; the value of the variable is not requested from the server but from the standby-server in redundant networks
RESOURCE	C	128	Resource 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	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
ADJWVBA	C	128	Linked 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 DEFINITION

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

Description	Type	Field size	Comment
AKTIV1	R	1	Limit value active (per limit value available)
GRENZWERT1	F	20	Technical value or ID number of a linked variable for a dynamic limit (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
HYSTERESE1	F	14	Hysteresis in %
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	Function linking
ASK_FUNC1	R	1	With interrogation before execution
FUNC_NR1	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
A_GRUPPE1	N	10	Alarm/event group
A_KLASSE1	N	10	Alarm/event class
MIN_MAX1	C	3	Minimum, Maximum
FARBE1	N	10	Color as Windows coding
GRENZTXT1	C	66	Limit value text
A_DELAY1	N	10	Time delay
INVISIBLE1	R	1	Invisible

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

7.5   Driver variables

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

- ▶   Information

- ▶   Configuration

- ▶   Statistics and

- ▶   Error messages

The definitions of the variables defined in the driver kit are available in the import file **drvvar.dbf** (on the CD in the directory: `CD_Drive:/Predefined/Variables`) and can be imported from there.

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

> ### 💡 Info
>
> *Not every driver supports all driver variants.*
>
> *For example:*
>
> ▷   Variables for modem information are only supported by modem-compatible drivers
>
> ▷   Driver variables for the polling cycle only for pure polling drivers
>
> ▷   Connection-related information such as ErrorMSG only for drivers that only edit one connection at a a time

INFORMATION

Name from import	Type	Offset	Description
MainVersion	UINT	0	Main version number of the driver.
SubVersion	UINT	1	Sub version number of the driver.
BuildVersion	UINT	29	Build version number of the driver.
RTMajor	UINT	49	zenon main version number
RTMinor	UINT	50	zenon sub version number
RTSp	UINT	51	zenon service pack number
RTBuild	UINT	52	zenon build number
LineStateIdle	BOOL	24.0	TRUE, if the modem connection is idle
LineStateOffering	BOOL	24.1	TRUE, if a call is received
LineStateAccepted	BOOL	24.2	The call is accepted
LineStateDialtone	BOOL	24.3	Dialtone recognized
LineStateDialing	BOOL	24.4	Dialing active
LineStateRingBack	BOOL	24.5	While establishing the connection
LineStateBusy	BOOL	24.6	Target station is busy
LineStateSpecialInfo	BOOL	24.7	Special status information received
LineStateConnected	BOOL	24.8	Connection established
LineStateProceeding	BOOL	24.9	Dialing completed
LineStateOnHold	BOOL	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 stopped
LineStateUnknow	BOOL	24.15	Connection status unknown
ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped

			For `driver stop`, the variable has the value `TRUE` and an **OFF** bit. After the driver has started, the variable has the value `FALSE` and no **OFF** bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

CONFIGURATION

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

Device	STRING	8	Name of the serial interface or name of the modem
ComPort	UINT	9	Number of the serial interface.
Baud rate	UDINT	10	Baud rate of the serial interface.
Parity	SINT	11	Parity of the serial interface
ByteSize	SINT	14	Number of bits per character of the serial interface Value = 0 if the driver cannot establish any serial connection.
StopBit	SINT	13	Number of stop bits of the serial interface.
Autoconnect	BOOL	16	TRUE, if the modem connection should be established automatically for reading/writing
PhoneNumber	STRING	17	Current telephone number
ModemHwAdr	DINT	21	Hardware address of current telephone number
RxIdleTime	UINT	18	Modem is disconnected, if no data transfer occurs for this time in seconds (s)
WriteTimeout	UDINT	19	Maximum write duration for a modem connection in milliseconds (ms).
RingCountSet	UDINT	20	Number of ringing tones before a call is accepted
ReCallIdleTime	UINT	53	Waiting time between calls in seconds (s).
ConnectTimeout	UDINT	54	Time in seconds (s) to establish a connection.

STATISTICS

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

PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

ERROR MESSAGES

Name from import	Type	Offset	Description
ErrorTimeDW	UDINT	2	Time (in seconds since 1.1.1970), when the last error occurred.
ErrorTimeS	STRING	2	Time (in seconds since 1.1.1970), when the last error occurred.
RdErrPrimObj	UDINT	42	Number of the PrimObject, when the last reading error occurred.
RdErrStationsName	STRING	43	Name of the station, when the last reading error occurred.
RdErrBlockCount	UINT	44	Number of blocks to read when the last reading error occurred.
RdErrHwAdresse	UDINT	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	SINT	25	Error message as number
DrvErrorMsg	STRING	30	Error message as text
ErrorFile	STRING	15	Name of error log file

8. Driver-specific functions

This driver supports the following functions:

With the defined communication parameters the driver establishes a connection to the Runtime system of the SoftPLC.

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

Parameters	Description
Drivers	Drop-down list with all drivers which are loaded in the project.
Current state	Fixed entry which has no function in the current version.
Driver commands	Drop-down list for the selection of the command.
▶ Start driver (online mode)	Driver is reinitialized and started.
▶ Stop driver (offline mode)	Driver is stopped. No new data is accepted. **Note:** If the driver is in offline mode, all variables that were created for this driver receive the status `switched off` (`OFF`; Bit 20).
▶ Driver in simulation	Driver is set into simulation mode. The values of all variables of the driver are simulated by the

	mode	driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▸	Driver in hardware mode	Driver is set into hardware mode. For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
▸	Driver-specific command	Enter driver-specific commands. Opens input field in order to enter a command.
▸	Activate driver write set value	Write set value to a driver is allowed.
▸	Deactivate driver write set value	Write set value to a driver is prohibited.
▸	Establish connection with modem	Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
▸	Disconnect from modem	Terminate connection (for modem drivers)
Show this dialog in the Runtime		The dialog is shown in Runtime so that changes can be made.

DRIVER COMMANDS IN THE NETWORK

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

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

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 7.00 -> Diagviewer.*

zenon driver log all errors in the log files. The default folder for the log files is subfolder `LOG` in directory `ProgramData`, example: `C:\ProgramData\zenon \zenon700\LOG` for zenon version 7.00 SP0. 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 currently created entries live

▶ customize the logging settings

▶ change the folder in which the log files are saved

Hints:

1. In Windows CE even errors are not logged per default due to performance reasons.

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

3. 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.

4. If you only use `Error logging`, the problem description is in column `Error text`. For other diagnosis level the description is in column `General text`.

5. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in `Error text` and/or `Error code` and/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.

6. 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.`

Info

You can find further information on the Diagnosis Viewer in the Diagnose Viewer (main.chm::/12464.htm) chapter.

10.2 Error numbers

Error code	Description
-1	Common ARTI error.
-101	The type of protocol is not supported.
-102	The communication channel is not valid or open.
-103	A channel with the specified parameters is already open.
-104	The wrong type of message was received.
-105	There was not any message received.
-106	There was not enough data for this message type received.
-201	There is too much data in the send queue.
-202	There is too much data in the receive queue.
-203	Only in the synchronous mode if last service has not finished yet.
-301	Common communication error in the system –dependent level.
-401	There wasn't such a file on the target or it could not be opened.
-402	The file data doesn't fit completely in the provided buffer.
-501	A strictly necessary function parameter is NULL or invalid.
-502	The maximum number of open channels is exceeded.
-503	There is no SDD assigned to the channel.
-504	There is no type table assigned to the SDD.
-505	The end of symbol table is reached.
-506	There is no symbol with that name found in the SDD.
-507	The data stream for reading the variables is bigger than the target's buffer.
-508	The data stream for writing the variables in the VarList is bigger than the target's buffer.
-509	Another VarList is still active with reading / writing its values.
-510	Writing various VarList blocks is not supported.

-511	The variable's swap size doesn't fit with the number of bytes to be written
-512	Error in parsing the symbol file.
-513	There is a new project on the runtime system so symbols data have changed.
-514	Error in sorting the symbol file.
-515	There is no project downloaded to the runtime system.

10.3   Check list

▶   Have you analyzed the error text file (which errors did occur)?

▶   Do the files "projectname_codesys.cmp" and "3S_Arti_drivername.dat" exist on the CE device?

Send the zenon project to support@copadata.com (mailto:support@copadata.com)