



**COPADATA**  
do it your way

# zenon driver manual

## BURPVI

v.8.00





©2018 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

<b>1. Welcome to COPA-DATA help .....</b>	<b>5</b>
<b>2. BURPVI.....</b>	<b>5</b>
<b>3. BURPVI - Data sheet .....</b>	<b>6</b>
<b>4. Driver history .....</b>	<b>8</b>
<b>5. Requirements.....</b>	<b>8</b>
5.1 PC .....	9
5.2 Control .....	9
<b>6. Configuration .....</b>	<b>9</b>
6.1 Creating a driver.....	10
6.2 Settings in the driver dialog .....	13
6.2.1 General .....	14
6.2.2 Global settings.....	18
6.2.3 Connections.....	21
6.2.4 Configuration file in redundant operation .....	40
<b>7. Creating variables.....</b>	<b>41</b>
7.1 Creating variables in the Editor.....	41
7.2 Addressing.....	45
7.3 Driver objects and datatypes .....	46
7.3.1 Driver objects .....	47
7.3.2 Mapping of the data types .....	48
7.4 Creating variables by importing .....	49
7.4.1 XML import.....	49
7.4.2 DBF Import/Export .....	50
7.4.3 Import PVI variables from the driver .....	55
7.5 Communication details (Driver variables).....	65
<b>8. Driver-specific functions .....</b>	<b>71</b>
<b>9. Driver command function .....</b>	<b>72</b>

<b>10. Error analysis.....</b>	<b>75</b>
10.1 Analysis tool .....	75
10.2 Check list .....	76
10.3 Error messages .....	77

# 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](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](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. BURPVI

Driver for B+R controls based on the PVI software from Bernecker+Rainer. The driver communicates with PVI. For example system 2000 family (2003, 2005 etc.), Acopos, X20 system, Automation PC, PowerPanel etc. ...

The driver supports spontaneous operation with hysteresis and online import of multidimensional arrays and arrays with a start index  $\neq 0$ .

Coupling: Serial, Ethernet

Protocol: PVI

The BUR-PVI driver replaces the BUR20032 driver. The addressing of these two drivers is not compatible with each other as the BUR20032 driver addresses the PVI items via an allocation file and the BUR-PVI saves the addressing directly via properties in the V-DLL. The driver is symbol-oriented.



### Attention

*Addressing of one-dimensional arrays:*

*If one-dimensional rays are addressed with a start index  $\neq 0$ , a comma must be attached to the index. For details see: Documentation B&R.*

*For example: zenon[3,]*

## 3. BURPVI - Data sheet

General:	
Driver file name	BURPVI.exe
Driver name	BR-PVI driver
PLC types	All Bernecker and Rainer PLCs which can use the PVI communication like System 2000 family (2003, 2005 etc), Acopos, X20 System, AutomationPC, PowerPanel and others.
PLC manufacturer	Bernecker + Rainer;

Driver supports:	
Protocol	PVI;
Addressing: Address-based	--
Addressing: Name-based	X
Spontaneous communication	X
Polling communication	X
Online browsing	X

Offline browsing	X
Real-time capable	--
Blockwrite	--
Modem capable	--
Serial logging	--
RDA numerical	X
RDA String	--
Hysteresis	X
extended API	X
Supports status bit <b>WR-SUC</b>	X
alternative IP address	--

Requirements:	
Hardware PC	RS 232 interface or Standard network card
Software PC	PVI software necessary, also under Windows CE. PC setup can be found on the installation media.
Hardware PLC	--
Software PLC	--
Requires v-dll	X

Platforms:	
Operating systems	Windows CE 6.0, Embedded Compact 7; Windows 7, 8, 8.1, 10, Server 2008R2, Server 2012, Server 2012R2, Server 2016;
CE platforms	x86;

## 4. Driver history

Date	Driver version	Change
1/26/2010	1400	Driver was created newly
1/14/2010	1500	Fixed error at the configuration
3/5/2010	1600	Adjusted to PVI 3.0
3/17/2010	1700	Revised dialogs

### DRIVER VERSIONING

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

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



#### Example

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

## 5. Requirements

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



## 5.1 PC

PVI software from Bernecker+Rainer.

## 5.2 Control

# 6. Configuration

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

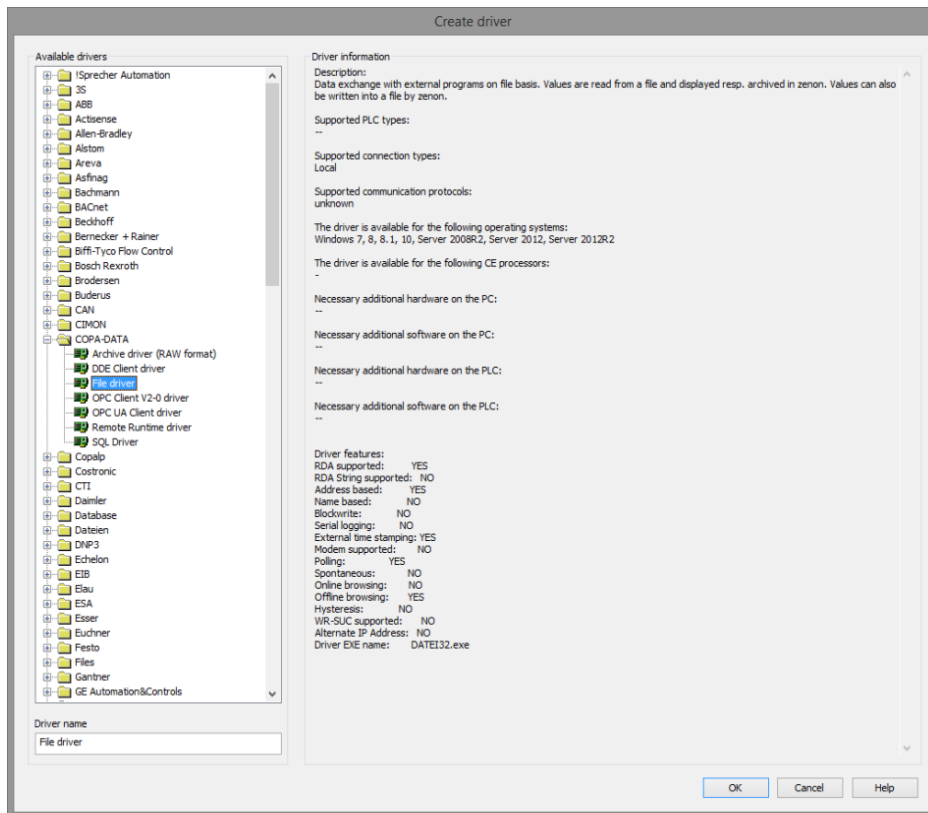


### Information

*Find out more about further settings for zenon variables in the chapter Variables (main.chm::/15247.htm) 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
<b>Available drivers</b>	<p>List of all available drivers.</p> <p>The display is in a tree structure:            [+] expands the folder structure and shows the drivers contained therein.            [-] reduces the folder structure</p> <p>Default: no selection</p>
<b>Driver name</b>	<p>Unique <b>Identification</b> of the driver.</p> <p>Default: Empty</p> <p>The input field is pre-filled with the pre-defined <b>Identification</b> after selecting a driver from the list of available drivers.</p>
<b>Driver information</b>	<p>Further information on the selected driver.</p> <p>Default: Empty</p> <p>The information on the selected driver is shown in this area after selecting a driver.</p>

## CLOSE DIALOG

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



### Information

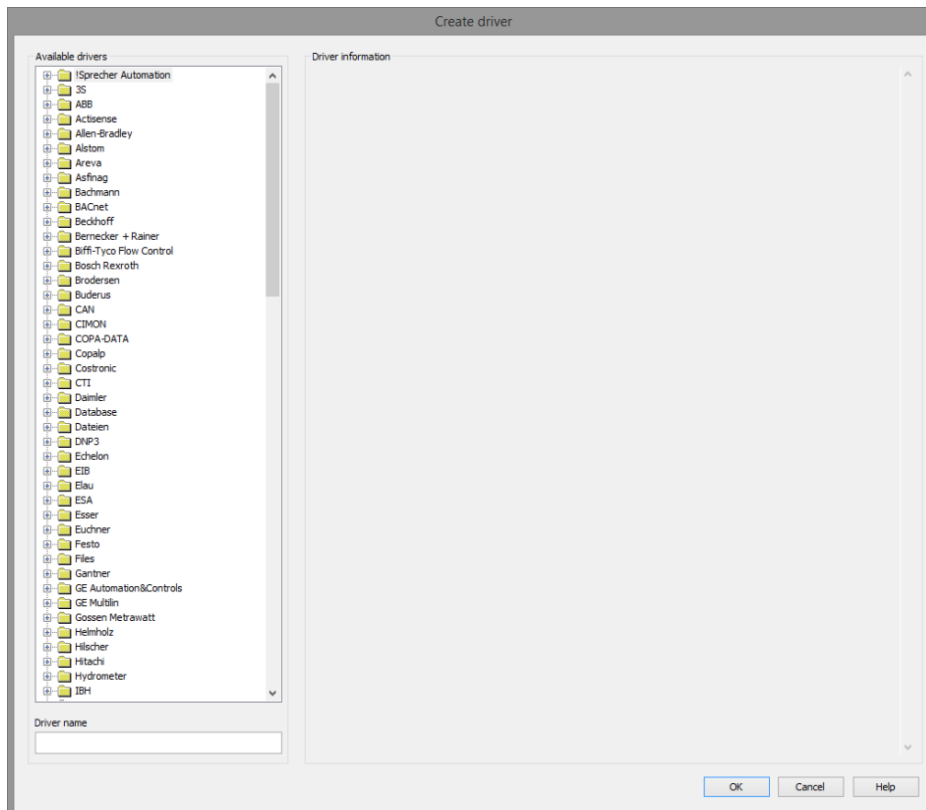
*The content of this dialog is saved in the file called `Treiber_[Language].xml`. You can find this file in the following folder: `C:\ProgramData\COPA-DATA\zenon[version number]`.*

## 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.

2. The dialog offers a list of all available drivers.



3. Select the desired driver and name it in the **Driver name** input field.  
This input field corresponds to the **Identification** property. The name of the selected driver is automatically inserted into this input field by default.

The following is applicable for the **Driver name**:

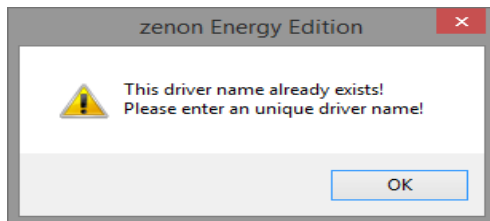
- The **Driver name** must be unique.  
If a driver is used more than once in a project, a new name has to be given each time.  
This is evaluated by clicking on the **OK** button. If the driver is already present in the project, this is shown with a warning dialog.
- The **Driver name** is part of the file name.  
Therefore it may only contain characters which are supported by the operating system.  
Invalid characters are replaced by an underscore (\_).
- **Attention:** This name cannot be changed later on.

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

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

## DRIVER NAME **DIALOG ALREADY EXISTS**

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



## ZENON PROJECT

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

- ▶ **Intern**
- ▶ **MathDr32**
- ▶ **SysDrv**



### Information

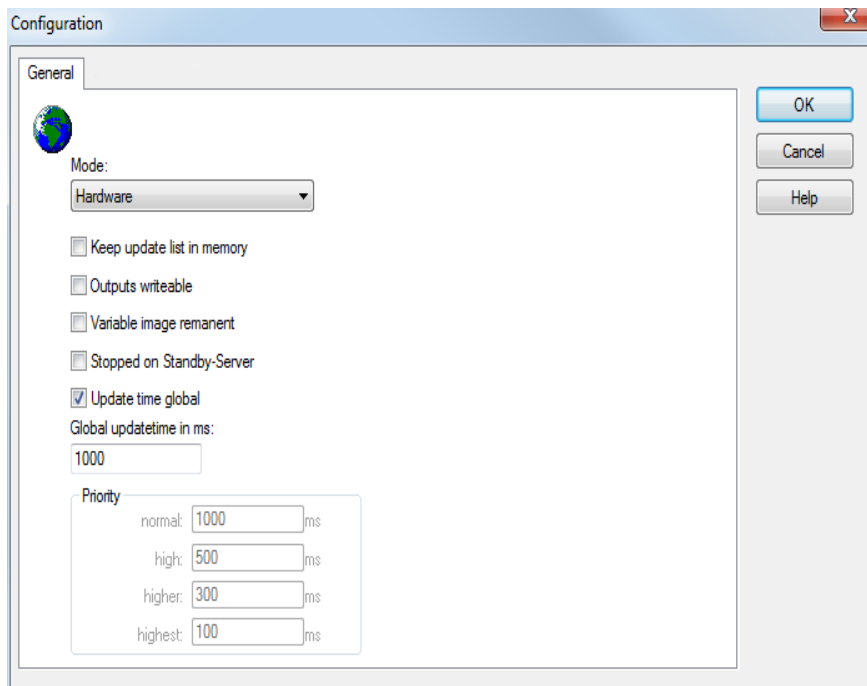
Only the required drivers need to be present in a zenon project. Drivers can be added at a later time if required.

## 6.2 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
<b>Mode</b>	<p>Allows to switch between hardware mode and simulation mode</p> <ul style="list-style-type: none"> <li>▶ <b>Hardware:</b> A connection to the control is established.</li> <li>▶ <b>Simulation - static:</b> 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.</li> <li>▶ <b>Simulation - counting:</b> 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.</li> <li>▶ <b>Simulation - programmed:</b> No 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).</li> </ul>
<b>Keep update list in the memory</b>	<p>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.</p>
<b>Output can be written</b>	<ul style="list-style-type: none"> <li>▶ <b>Active:</b> Outputs can be written.</li> <li>▶ <b>Inactive:</b> Writing of outputs is prevented.</li> </ul> <p><b>Note:</b> Not available for every driver.</p>
<b>Variable image remanent</b>	<p>This option saves and restores the current value, time stamp and the states of a data point.</p> <p>Fundamental requirement: The variable must have a valid value and time stamp.</p> <p>The variable image is saved in mode hardware if:</p>

- ▶ 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.



<b>Stop on Standby Server</b>	<p>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.</p> <p><b>Attention:</b> If this option is active, the gapless archiving is no longer guaranteed.</p> <p>► <b>Active:</b> 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 <b>switched off (statusverarbeitung.chm::24150.htm)</b> but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.</p> <p>Default: <i>Inactive</i></p> <p><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.</p>
<b>Global Update time</b>	<p>Setting for the global update times in milliseconds:</p> <p>► <b>Active:</b> The set <b>Global update time</b> is used for all variables in the project. The priority set at the variables is not used.</p> <p>► <b>Inactive:</b> The set priorities are used for the individual variables.</p> <p><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.</p>
<b>Priority</b>	<p>The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.</p> <p>The variables are allocated separately in the settings of the variable properties.</p> <p>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.</p> <p><b>Attention:</b> Priority classes are not supported by each driver, e.g. spontaneously communicating zenon drivers.</p>

**CLOSE DIALOG**

Option	Description
<b>OK</b>	Applies all changes in all tabs and closes the dialog.

<b>Cancel</b>	Discards all changes in all tabs and closes the dialog.
<b>Help</b>	Opens online help.

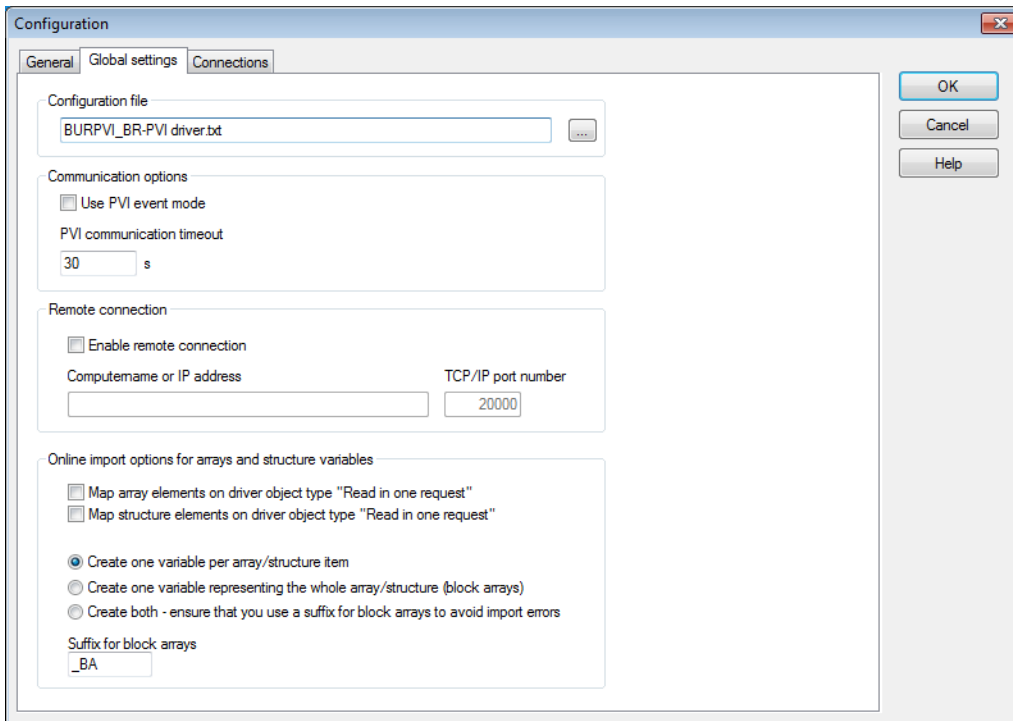
## UPDATE TIME FOR SPONTANEOUS DRIVERS

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 Global settings

General settings for the PVI communication are set in the **Global Settings** tab. These settings are true for all connections which are available in the driver.



The screenshot shows the 'Configuration' dialog box with the 'Global settings' tab selected. The dialog has three tabs: 'General', 'Global settings', and 'Connections'. The 'Global settings' tab contains the following fields and options:

- Configuration file:** A text box containing 'BURPVI\_BR-PVI driver.txt' and a 'Browse...' button.
- Communication options:**
  - ☐ Use PVI event mode
  - PVI communication timeout:** A text box containing '30' and a unit 's'.
- Remote connection:**
  - ☐ Enable remote connection
  - Computename or IP address:** A text box.
  - TCP/IP port number:** A text box containing '20000'.
- Online import options for arrays and structure variables:**
  - ☐ Map array elements on driver object type "Read in one request"
  - ☐ Map structure elements on driver object type "Read in one request"
  - ☒ Create one variable per array/structure item
  - ☐ Create one variable representing the whole array/structure (block arrays)
  - ☐ Create both - ensure that you use a suffix for block arrays to avoid import errors
  - Suffix for block arrays:** A text box containing '\_BA'.

On the right side of the dialog, there are three buttons: 'OK', 'Cancel', and 'Help'.

Parameter	Description
<b>Configuration file</b>	<p>File in which the configuration of connections and the global settings are saved. Click on ... in order to open the Windows Explorer.</p> <p>Note the hints in chapter: Configuration file in redundant operation (on page 40).</p>

## REMOTE CONNECTION

Parameter	Description
<b>Use PVI event mode</b>	<p><b>Active:</b> All PVI variables are requested via event mode (PVI command <code>AT=rwe HY=0</code>).</p> <p><b>Inactive:</b> PVI variables are read cyclically with the setting of the global update time (<b>AT=rw, RF=global update time in ms</b>).</p>
<b>PVI communication timeout [s]</b>	<p>Timeout of PVI communication in seconds.</p> <p>PviInitialize (dwTimeout, ...)</p>
<b>Enable Remote Connection</b>	<p><b>Active:</b> Connection to a remote PVI system via TCP/IP is established. The communication takes place exclusively via this gateway. A local PVI installation is always necessary.</p> <p>In addition to the settings for the driver, you must configure the setting for the remote communication via TCP/IP on the PVI itself (PVI client/Manager) - locally and on the remote system.</p>
<b>Computername or IP address</b>	<p>Computer name or IP address on which the remote PVI system is installed and runs.</p>
<b>TCP/IP Port number</b>	<p>TSP/IP port number of the remote PVI system.</p> <p><b>Note:</b> Any firewall that may be present must be configured so that the target port can be reached.</p>

## ONLINE IMPORT OPTIONS FOR ARRAYS AND STRUCTURE VARIABLES

Options for the online import of array variables and structure variables.

Parameter	Description
Map array elements on driver object Type "Array as whole"	Active: At the import of array elements select <b>Driver object typesRead in a request</b> - exception for arrays in structures.
Map structure elements on driver object Type "Array as whole"	Active: At the import of array elements select <b>Driver object typesRead in a request</b> - also for arrays in structures.
Create one variable per array/structure item	Active: Only individual variables are created and no nodes are displayed. Default setting.
Create one variable representing the whole array/structure (block arrays)	Active: For arrays and structures, only block variables are created, but no variables that represent individual variables of structures/arrays. This option has no influence on top level variables.
Create both - ensure that you use a suffix for block arrays to avoid import errors	Active: For arrays and structures, both individual elements and blocks that represent the whole structure are represented.
Suffix for block arrays	Suffix for the zenon names of variables that were created as block array - variables that represent a whole structure or complete array.

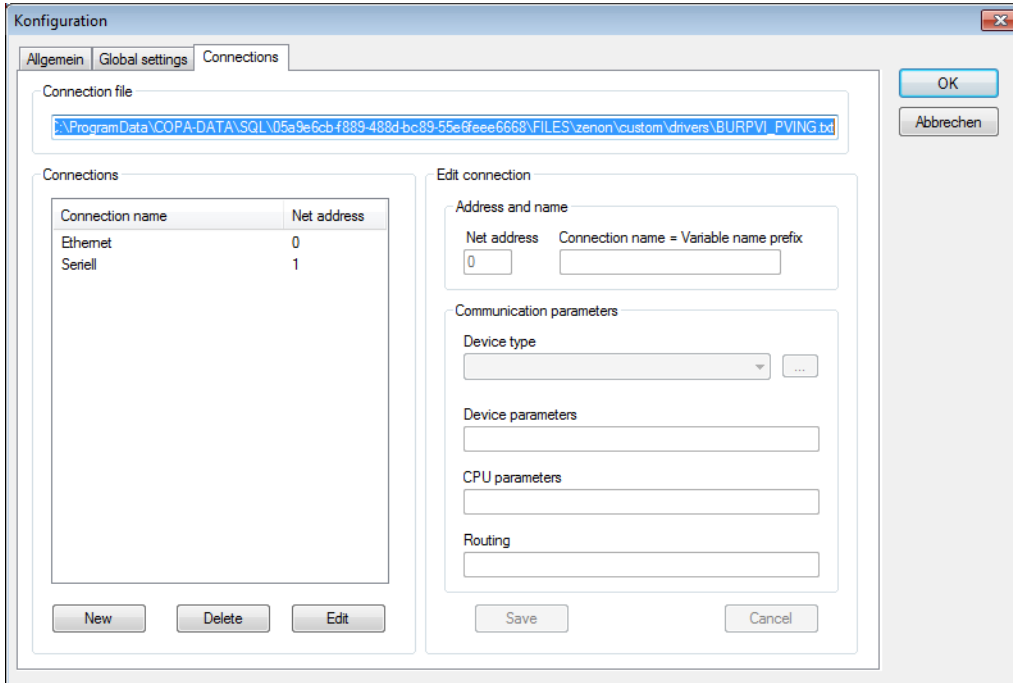
## ARRAY AS A WHOLE

With this all elements of the driver object type **Array as whole** are mapped. All elements of this object type are no longer read in the event that a value changes, but polled cyclically. The driver always reads in the whole array as a block.

A block array is a large variable in zenon that reads a complete block from the PLC. Many individual arrays are therefore read as a single array.

## 6.2.3 Connections

On this tab you carry out the settings for the connection. You can create any number of connections in the driver. All connections use the global settings which are defined on tab **General Settings** (on page 18).



The screenshot shows the 'Konfiguration' dialog box with the 'Connections' tab selected. The 'Connection file' field contains a long path. The 'Connections' list shows two entries: 'Ethernet' with net address '0' and 'Seriell' with net address '1'. The 'Edit connection' panel is active, showing fields for 'Address and name', 'Communication parameters', 'Device type', 'Device parameters', 'CPU parameters', and 'Routing'. The 'OK' button is labeled 'OK' and the 'Abbrechen' button is labeled 'Abbrechen'.

Connection name	Net address
Ethernet	0
Seriell	1

**Edit connection**

**Address and name**

Net address:  Connection name = Variable name prefix:

**Communication parameters**

Device type:

Device parameters:

CPU parameters:

Routing:

Buttons: New, Delete, Edit, Save, Cancel, OK, Abbrechen

Parameter	Description
<b>Connection file</b>	Storage location for the configuration file in which the connection and the global setting is saved. The file is set in tab <b>General Settings</b> (on page 18).
<b>Connections</b>	List of available connections.
<b>New</b>	Adds new entry to the list. Settings are carried out in the field to the right of the list.
<b>Delete</b>	Deletes selected entry from the list.
<b>Edit</b>	Makes it possible to configure the selected entry. The area to the right of the list for editing the connection is activated. <b>Attention!</b> The driver dialog cannot be closed in editing mode. Only once the editing mode has been left using <b>Save</b> or <b>Cancel</b> can you close the driver dialog.
<b>Edit connection</b>	Edit connection settings.
<b>Address and name</b>	Address and name.
<b>Net address</b>	Net address of the connection for the allocation of variables. <b>Attention:</b> If you change the net address here, you must also change the net address of all other variables of the connection. Otherwise the allocation is no longer correct and variables cannot communicate in the Runtime!
<b>Connection name</b>	Symbolic connection name. This name is used as a prefix for the variable name.
<b>Communication parameters</b>	Communication parameter:
<b>Device type</b>	Selection of the PVI device type from the drop-down list: <ul style="list-style-type: none"> <li>▶ Serial</li> <li>▶ Ethernet</li> <li>▶ Modem</li> <li>▶ CAN</li> <li>▶ Shared</li> <li>▶ User defined</li> </ul> For types <b>Serial</b> and <b>Ethernet</b> you can open a dialog for configuration by clicking on ....
<b>Device parameters</b>	Direct input of the PVI device parameter. If you carry out the setting for the connection with the configuration dialog for Ethernet or Serial, the settings are overwritten here.  Ethernet default: /IF=TCPIP /SA=1  Serial default: /IF=COM1 /BD=57600 /PA=2 /IT=1  You can find the settings for the single communication types in

	chapter Device parameter - INA2000 device object (on page 31)
<b>CPU parameters</b>	Direct input of the PVI CPU parameter. This is the station address of the CPU. For serial connections no station address is needed.  Ethernet default: /DA=2  Serial default: -
<b>Routing</b>	Direct input of a routing path for the routing via PLC stations. See description for routing further down.
<b>Save</b>	Saves the configuration of the selected connection.
<b>Cancel</b>	Discards configuration.
<b>OK</b>	Saves all changes and closes dialog.
<b>Cancel</b>	Discards all changes and closes the dialog.



### Information

*Maximum number of Ethernet connections: 256.*

## ROUTING

With the help of routing, communication connections can be established via a PLC station to PLC stations of other networks. This connections can run via several PLC stations. The single stations are defined in the routing path.

### SYNTAX

<Entry 1>/<Entry 2>/.../<Entry n>

The single path entries are separated by character '/'.

Syntax for an entry:

SL<Slot>.SS<Subslot>.IF<Interface>.<Address>

All partial entries are separated by character '.'. Statement SL<Slot> and SS<Subslot> need not to be specified for 0 or 1. The statement <Address> is only needed for connections via the network.

Parameter	Description
SL<Slot>	Slot number: 0 (1) - 15.
SS<Subslot>	Subslot number: 1 - 3.
IF<Interface>	Interface number: 1 - 15.
<Address>	Station address: 1 - ff (hexadecimal).

Routing on or via peripheral processors:

Parameter	Description
CP	Main processor.
PP<Slot>	peripheral processor, slot number: 0 (1) - 15.

## EXAMPLES

/CN=IF3/IF2.7

From the PC to the first PLC, from there via IF3 to the next station (PLC) and then via IF2 to a CAN station with station number 7.

/CN=SL6.IF1

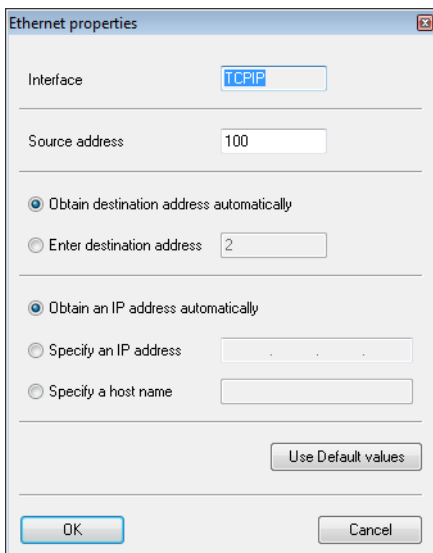
From the PC to the first PLC then on via slot 6 (e.g. IF 050) and IF1

/CN=PP3

From the PC to the first PLC and on to the peripheral processor 3

## ETHERNET CONNECTION

A click on ... in area **Device type** opens the dialog for the configuration of the Ethernet connection if you have selected **Ethernet**:



Ethernet properties

Interface: TCP/IP

Source address: 100

☒ Obtain destination address automatically  
☐ Enter destination address: 2

☒ Obtain an IP address automatically  
☐ Specify an IP address:   
☐ Specify a host name:

Use Default values

OK Cancel

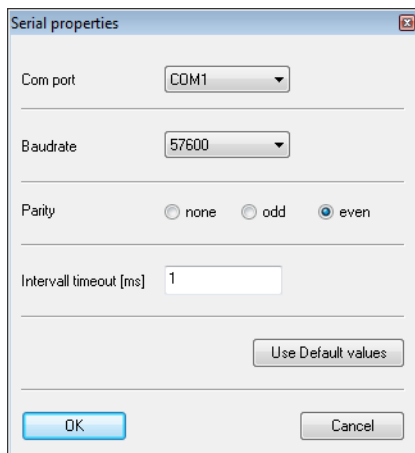


Parameter	Description
<b>Interface</b>	Defines the interface. Must always be TCP/IP.
<b>Source address</b>	<p>Station number of the source station (own station).</p> <p><b>Warning!</b> The station number must be unique within the INA2000 network. As default B&amp;R Automation Studio uses station number 1. Thus it can easily happen that the communication to the PLC does not work if station number 1 is used. The highest possible station number is 99 for SG3 and 255 for SG4. See also Configuration file in redundancy operation (on page 40).</p>
<b>Obtain destination address automatically</b>	Obtain destination address automatically: PVI connects to the defined device (IP address or computer name must be set) and determines the station address automatically.
<b>Enter destination address</b>	Enter destination address: Enter the station address of the target system. If <b>Obtain an IP address automatically</b> is activated, a broadcast is carried out in the network in order to find the station.
<b>Obtain an IP address automatically</b>	Obtain an IP address automatically: Carries out a broadcast in the network and tries to find the station with the station address defined at <b>Enter destination address</b> .
<b>Specify an IP address</b>	Define IP address: Enter the IP address of the station.
<b>Specify a host name</b>	Define host name: Enter the host name of the station.
<b>Use Default values</b>	Resets the properties to the default settings.
<b>OK</b>	Applies settings and closes the dialog.
<b>Cancel</b>	Discards settings and closes the dialog.

As an alternative you can enter the values in areas **Device Parameter**, **CPU Parameter** and **Path target system**.

## SERIAL CONNECTION

A click on ... in area **Device type** opens the dialog for the configuration of the serial connection if you have selected **Serial**:



Parameter	Description
<b>Com port</b>	Defines interface. Default: COM1
<b>Baud rate</b>	Defines baud rate. Default: 57600
<b>Parity</b>	Defines parity: <ul style="list-style-type: none"> <li>▶ no</li> <li>▶ odd</li> <li>▶ even</li> </ul> Default: even
<b>Interval timeout (ms)</b>	Defines the interval in milliseconds. Default: 1
<b>Use Default values</b>	Resets the properties to the default settings.
<b>OK</b>	Applies settings and closes the dialog.
<b>Cancel</b>	Discards settings and closes the dialog.

As an alternative you can enter the values in areas **Device Parameter**, **CPU Parameter** and **Path target system**.

## INA2000 communication

The driver supports the communication in accordance with the INA2000 protocol.  
The INA2000 communication is supported by the following devices:

- ▶ Serial communication
- ▶ CAN communication
- ▶ Ethernet UDP communication
- ▶ Modem communication
- ▶ Profibus FDL communication
- ▶ LS251 communication
- ▶ AR000 communication

Each INA2000 communication possesses a life monitoring. As soon as a connection is established, it is checked each second by exchanging small data packages (ping/pong).

#### **GENERAL PLC PREREQUISITES:**

The INA2000 communication works limited (services) as of PLC operating system 1.91. To be able to use all services (e.g. upload, download of modules), PLC operating system 2.10 is necessary.

If multimaster abilities or a TCP/IP communication between PC and PLC are necessary, you must use PLC operating systems 2.20 or higher. For a INA2000 event handling via the PLC (PLC monitors data change), a PLC operating system 2.24 or higher is needed.

INA2000 orders are processed in the PLC in the remaining time. To receive suitable response times, a balance time of at least 20% is required. If 20% balance time cannot be guaranteed due to application reasons, longer answer times will occur and therefore the data will be refreshed slower.

#### **SERIAL COMMUNICATION**

The serial communication can only be operated as point-to-point connection (RS232 or RS422).

#### **CAN COMMUNICATION**

The CAN communication is used as INA2000 network with a maximum of 32 (extendable to 255) stations. The individual stations are distinguished by the CAN node number.

#### **PC REQUIREMENTS**

Please refer to the manufacturers description to find out which CAN card can be used with which Windows system.

## PLC REQUIREMENTS

To allow an INA2000 communication via CAN, you must activate CAN communication in Automation Studio. See Automation Studio documentation (*Project -> Hardware configuration -> CAN configuration*).

## ETHERNET UDP COMMUNICATION

### PC REQUIREMENTS

The TCP/IP protocol must be installed.

### PLC REQUIREMENTS

To allow an INA2000 communication via Ethernet, you must activate Ethernet communication in Automation Studio. See Automation Studio documentation (*Project -> Hardware configuration -> Ethernet configuration*).

For SG4 (i386):

The configuration of the IP address and the station number (INA node number) takes place in the Ethernet property dialog of the Automation Studio. No additional modules must be transferred to the PLC.

For SG3 (m68k):

For INA2000 communication modules "FBTCPIP.BR" and "TCIPMGR.BR" must be burned. In addition a data object "TCIPCFG.DAT" with the corresponding settings must exist.

### USE SEVERAL ETHERNET NETWORK CARDS AT THE PC.

How to make sure that PVI uses the correct network card in order to communicate with the PLC when two or more network cards are available in the PC.

In order for the correct network card to be selected for the PLC communication, the subnet mask must be set correctly. The IP address of the target station (PLC) is linked with the subnet mask with AND; thus the correct network card is found and used for the communication.

Example:

Device parameter: /IF=tcPIP /SA=1

CPU parameter /DA=13 /DAIP=172.43.70.13

IPAdr NW1: 172.43.71.12 subnet mask: 255.255.255.0

IPAdr NW2: 172.43.70.12 subnet mask: 255.255.255.0

IPAdr PLC: 172.43.70.13

With this network card two is used for the communication to the PLC. If a DHCP server is configured for a network card, you must know which IP addresses and subnet masks the DHCP server assigns so that you can configure the second network card correctly.

## MODEM COMMUNICATION

The Modem communication just as the serial communication is a point-to-point connection.

Modem features:

- ▶ Transparency: The modem is accessed only via standard Windows functions. I.e. any modem can be used which can be installed under Windows. The user must only take care about the installation of the modem. He does not need any special knowledge about the modem (AT commands, etc.).
- ▶ Automatic connection reestablishment: An actively established connection (see below) will be monitored constantly. When the connection is lost (e.g. interruption of the phone line, unplugging the modem, turning of the modem), a new connection establishment is tried in periodic intervals. The number of tries and the interval can be engineered.
- ▶ Active connection establishment: The Windows computer establishes a dial connection to the stated phone number and transfers the protocol defined by PVI line on this connection.
- ▶ Passive connection establishment: The Windows computer waits for an incoming call, picks up automatically and establishes a connection. With this operation mode it is for example possible that a PLC itself initiates a connection establishment to a Windows computer via PVI.

## PC REQUIREMENTS

The PVI modem device needs Microsoft TAPI version 2.0. This version is already installed at Windows NT 4.0. When using Windows 95 it is necessary that the default TAPI version 1.4 is replaced by a newer version. The most topical Microsoft ([www.microsoft.com](http://www.microsoft.com)) version available is version 2.1. To install TAPI version 2.1 under Windows 95, start program "tapi2195.exe" in subfolder "Pvi\SysSetup\Modem". This file is installed via PVI setup option "Modem system components".

The installation of a modem is carried out in Windows via control panel -> modem. If the dialog is opened it is possible to add new modems and to change the settings of the modem ("Properties"). The label of the modem in this dialog matches the modem name which must be stated as PVI device parameter (/MO).

The behavior during dialing can be configured under "Control panel -> Phone" (current location, dialing card, etc.).

## PROFIBUS FDL COMMUNICATION

Attention:

The INA2000 communication via Profibus FDL is only possible on System 2000. External systems are not supported by the PVI!

The settings of the PLC Profibus card (NW100/NW150) and the PC card (5A1104.00-090) can be taken from the PROFIBUS user manual (MASYS2PB-0).

## **PC REQUIREMENTS**

The data exchange with the PC takes place via a 32 KB DPR. For this a free address area must be defined on the PC. On the Profibus card this is set with two hex switches SW1 and SW2 (SW1 = 0; SW2=D). If only one Profibus card is used in the system, jumper BR1 and BR2 must be let open. Additional information can be found in the PROFIBUS user manual.

For this memory to be deallocated on the PC, the memory area must be excluded in the BIOS. On the IPC5000 in the BIOS under "PNP/PCI Configuration" you must exclude either 32 KB (1 Profibus card) or 64 KB (for 2 Profibus cards) starting at basic address D000.

## **PLC REQUIREMENTS**

NW100 or NW150 with revision higher/equal to xx.05.

On the PLC operating system 2.00 or higher is necessary. For the INA2000 communication module "FBPB.BR" must be burnt. The Profibus configuration module is already part of the operating system and can be changed with the help of suitable tools if necessary.

To allow an INA2000 communication via Profibus, you must extend the configuration for the INA2000 Profibus communication in the SYSCONF module.

## **LS251 COMMUNICATION**

### **PC REQUIREMENTS**

With Windows the LS251 card is recognized by the operating system automatically after it has been plugged in. In the hardware wizard you can select the corresponding driver under "Pvi\Drivers\LS251\W2k\_xx".

Runtime PLC prerequisites:

On LS251 PLC operating system V2.01 or higher must be installed.

## **AR000 COMMUNICATION**

Communication with AR000 takes place via an Ethernet UDP device and an Ethernet UDP station with local IP address 127.0.0.1.

Connection description for device: "/IF=Tcplp", Connection description for station: "/DAIP=127.0.0.1/REPO=11160".

## PC REQUIREMENTS

The AR000 Runtime emulation must be installed.

## Device parameter - INA2000 device object

With the INA2000 device object the used communication device is defined. You must enter the parameters described here in the connection configuration in field **Device parameter**.

The following communication parameters can be defined:

- ▶ Serial device
- ▶ CAN device
- ▶ Ethernet UDP device
- ▶ Modem device
- ▶ Profibus FDL device
- ▶ LS251 device

The communication device including the necessary device parameters are defined in the connection description of the device object. Within the connection description the single parameters are distinguished by parameter identifications. The parameter identification always starts with character '/'.

## SYNTAX OF THE CONNECTION DESCRIPTION

```
/IF=<device name>  [/<identification1>=<parameter value>  [/<identification2>=<parameter value> ...
]]
```

Parameter /IF is the same for all communication devices. All other parameters depend on the used communication device. The device name is not case-sensitive. You must insert at least one space character between the parameter declaration.

In the object description the connection description must always be between quotation marks ("...").

## EXAMPLE FOR CONNECTION DESCRIPTION

```
CD="/IF=com1 /BD=57600 /PA=2"
```

## SERIAL DEVICE

The following table shows all definable parameters defined in the connection description of the device object for serial communication.

Parameters	Values	Input	Description
/IF	com1 ...comX	None	by default: com1 to com4. With corresponding serial interface cards or adapters even more than com4 is possible. Example: "/IF=com1".
/BD	9600, 19200, 38400, 57600, 115200	57600	Baud rate in bits per second.
/RS	-1, 0, 232, 422, 485	232	set protocol RS232 or RS422. RS485 is not supported by INA2000. Example: "/RS=422".  With this parameter the flow control of the line CTS (clear-to-send) and RTS (request-to-send) of the serial interface is set.  Possible parameter values:  = -1: PVI does not change the current setting (see properties interface device) of the CTS/RTS flow control. = 0: always switch off RTS line (RTS flow control is deactivated). = 232: Use RTS handshake. = 422: Always switch on RTS line. = 485: Switch on RTS line in order to trigger a transfer and during a transfer.  The CTS flow control is deactivated for all parameters $\geq 0$ .  Some USB/serial adapter (USB to serial interface) cannot handle the CTS/RTS flow control correctly. To establish an RS232 connection via these adapters, parameter "/RS=0" must be entered.
/PA	0 - 4	1	Setting for parity.  0=NOPARITY, 1=ODDPARITY, 2=EVENPARITY, 3=MARKPARITY, 4=SPACEPARITY.  Default setting for INA2000 communication is "/PA=2".



### Example

Example for connection description:

CD="/IF=com2 /BD=115200 /PA=2"



## CAN DEVICE

To communicate with a INACAN device, you must first set it up as CAN device via B&R device configuration. The configuration is started via control panel and CAN device. The device list of the configuration shows all already setup CAN devices. The device number must also be entered in the PVI device name. At this device CAN1 matches the PVI device name INACAN1; CAN2 matches INACAN2 and so on. With the B&R CAN device configuration you also set the device resources (IRQ, port address, etc.).

The following table shows all definable parameters defined in the connection description of the device object for CAN communication.

Parameters	Values	Input	Description
/IF	inacan1 ... inacanX	None	CAN device. The used device must be entered in the device list of the B&R CAN device configuration. Example: "/IF=inacan1".
/CNO	0, 1	0	Number (channel) of the CAN controller.  On the LS172 card 2 CAN controllers are available. With parameter /CNO the desired controller is selected. For the standard CAN controller you must not enter another values as 0 (zero).  Example for LS172 card:  CAN-Bus 1: "/IF=inacan2 /CNO=0", CAN-Bus 2: "/IF=inacan2 /CNO=1".
/IT	0 - 60000, 0 = off	0	Interval timeout (ms).  Defines the maximum time which may pass between receiving and sending an INA frame between two CAN messages. This parameter is used together with parameter /RT of the INA2000 CPU objects in order to recognize a connection termination.  The parameter must not be smaller than the reaction time of the control (5 - 30 ms) plus a cushion of at least 25 ms (if there is a high interrupt strain on the PC, it must be respectively higher) but it should be smaller than /RT. If no quick recognition of the connection termination is necessary, the monitoring of the interval timeout can be switched off ("/IT=0").  Example: "/IT=80".
/BI	0 - 2047 / 536870911	1598	Basic CAN-ID of the INA2000 communication.  All stations of the INA2000 network must have the same setting. Example: "/BI=1598".
/MDA	32 - 255	32	Number of maximally possible INA2000 stations (=highest station number).  All stations of the INA2000 network must have the same setting. Example: "/MDA=50".
/SA	1 - /MDA	1	Station number of the source station (own station).  The station number must be unique within the INA2000 network. Example: "/SA=3".
/BD	10000, 20000, 50000, 100000,	*)	Baud rate (data rate) at the CAN bus in bits per second. The rate can also be in Kbits per second.  All stations of the INA2000 network must have the same setting. Example: "/BD=250000" or "/BD=250".

	125000, 150000, 250000, 500000, 800000, 1000000		
/CMODE	11, 29	*)	<p>CAN communication with 29 bit identifier (extended frames) or with 11 bit identifier (standard frames).</p> <p>If 29 bit CAN identifiers (extended frames) are used, 11 bit identifiers cannot be received or sent.</p> <p>All stations of the INA2000 network must have the same setting.</p> <p>Example: "/CMODE=29".</p>
/CT	>= 1, 0 = off	*)	<p>Cycle time (in ms); in this time not more than the stated number of CAN messages (parameter /MC) can be sent.</p> <p>Values &lt; 20 ms make no sense.</p> <p>Example: "/CT=20".</p>
/MC	>= 1	*)	<p>Maximum number of CAN messages which are sent in the states cycle time (parameter /CT).</p> <p>Example: "/MC=15".</p>

\*) Parameters /BD, /CMODE, /CT and /MC can also be set with the B&R CAN device configuration. If these parameters are not stated in the connection description (= recommended method), the values set in the configuration are used. If one of the parameters is stated in the connection description, the respective configuration setting is overwritten.

Parameters /CT and /MC serve as message limitation. With this the number of sent CAN messages per cycle time can be limited. The traffic caused by the PC on the CAN bus is reduced by this. The message limitation is important if other bus members can only process a certain number of received CAN messages at a fixed baud rate or reduce the interrupt strain of other bus members in general. The disadvantage of the message limitation is a slower CAN communication.

For the INA2000 communication each station needs 3 CAN-IDs. The CAN-IDs are created from the basic CAN-ID (parameter /BI), the station number (node number) and the maximum number of stations (parameter /MDA):

- ▶ ID1 (initiate request) = <Basic CAN-ID> + <station number> - 1
- ▶ ID2 (initiate response) = <Basic CAN ID> + <station number> - 1 + <maximum number of stations> \* 2

- ID3 (data segment) = <Basic CAN ID> + <station number> - 1 + <maximum number of stations>



### Example

Example for connection description:

CD="/IF=inacan3 /CNO=1 /SA=3"

## ETHERNET UDP DEVICE

The following table shows all definable parameters defined in the connection description of the device object for Ethernet UDP communication.

Parameter s	Values	Input	Description
/IF	tcpip	None	Ethernet UDP device. Statement: "/IF=tcpip".
/LOPO	1024 - 32767	11159	Port number of the source station (own station).  If the value is stated in hexadecimal, "0x" must be placed in front (e.g. "/LOPO=0x2b97").  You must only set another port number as the default if it is not unique within the local computer.  Example: "/LOPO=11159".
/SA	0 - 99 / 255	1	Station number of the source station (own station).  The station number must be unique within the INA2000 network. The highest possible station number is 99 for SG3 and 255 for SG4.  Example: "/SA=3".



### Example

Example for connection description:

CD="/IF=tcpip /SA=3"

## MODEM DEVICE

The following table shows all definable parameters defined in the connection description of the device object for modem communication.

Parameters	Values	Input	Description
/IF	modem1 ... modemX	None	Modem device.  Example: "/IF=modem1".
/MO	Modem description	None	Description of the modem as in the setup dialog (control panel -> modem). The string must be between single quotation marks. If a single quotation mark should be used within a string, you must use two single quotation marks (e.g. m'56k is stated as /MO="56k').  Example: "/MO='MicroLink 56k'".
/TN	Phone number	None	Phone number. The phone number which should be called in accordance with "ITU-T Recommendation E.123", z.B. +43(7748)6586. You must always enter the complete phone number (including international access code); the conversion to the actual phone number is carried out automatically. This string must be between single quotation marks. If it should be waited for a call, you must enter an empty string (/TN=").  Example: "/TN='+43(7748)6586'".
/MR	0 - INFINITE	INFINITE	Maximum number of failed re-dialing attempts. Defines the number of tries a failed connection should be tried to be reestablish. If you enter INFINITE the number of retries is unlimited. If you enter 0, there is no retry.  Parameter /MR only has a meaning if /TN is not empty.  Examples: "/MR=50", "/MR=INFINITE".
/RI	0 - 3600	60	Time interval between retries in seconds. If a retry fails, this time is waited until a new try is started. Parameter /RI only has a meaning if /TN is not empty.  Example: "/RI=120".
/IT	0 - 60000	40	Interval timeout (ms).  Defines the maximum time which may pass between the reception of two successive characters.

			Example: "/IT=100".
--	--	--	---------------------

In addition to parameter /IF, you must also always enter parameter /MO and /TN.

Reading from the serial interface is carried out by an operating system function. A buffer is handed over to this function. In this buffer the received data is stored. The operating system finishes the reading if one of the following situation occurs:

1. The buffer is full.
2. Since receiving the last character a time interval which is larger than the timeout interval has past.

The correct setting of parameter /IT for the timeout interval is therefore especially important. If the timeout interval is too small, INA2000 frames could be lost. This would lead to a connection termination. If the timeout interval is too large, the data throughput suffers as the timeout interval passes before the PC recognizes a received frame. Default value 40 ms has been chosen because of measurements which show that a modem (by internal buffering) adds 35 ms breaks in the byte stream. Normally it should not be necessary to select another value. We can however not rule out that you must set a higher value if using another modem type or bad line quality (modems have their own transfer security with automatic retransmission).



### Example

Example for connection description:

```
CD="/IF=modem1 /MO='ZyXEL MODEM Omni 288S' /TN='+43(7748)999'"
```

## PROFIBUS FDL DEVICE

The following table shows all definable parameters defined in the connection description of the device object for Profibus FDL communication.

Parameters	Values	Input	Description
/IF	pbusfdl1 ... pbusfdIX	None	Profibus device.  Example: "/IF=pbusfdl1".
/BA	0x00000 - 0xFFFFF	0xD0000	Basic address of the DPR of the Profibus card. If the value is stated in hexadecimal, "0x" must be placed in front.  Example: "/BA=0xD0000".
/FF	Path name*)	nw_load.bin	Path name of the firmware file for the Profibus card.  Example: "/FF=nw.bin".
/FC	Path name*)	nw_pb_32.br	Path name of the network configuration file for the Profibus card.  Example: "/FC=nw_pb.br".
/CB	1 - 255	2	Number of communication buffers of the Profibus card.  Example: "/CB=4".
/SA	0-127	1	Station number of the source station (own station).  The station number must be unique within the INA2000 network.  Example: "/SA=2".

\*) A standard firmware file and configuration file is part of the PVI. If no path is entered, the files must be in the folder of the PVI manager.

Settings such as baud rate and timeout cannot be defined via the device parameters. For this the configuration file must be changed with the right tools.



### Example

Example for connection description:

```
CD="/IF=pbusfdl1 /FF=c:\pbconfig\nw_load.bin /FC=c:\pbconfig\nw_pb_32.br /SA=1"
```

## LS251 DEVICE

The following table shows all definable parameters defined in the connection description of the device object for LS251 communication.

Parameters	Values	Input	Description
/IF	Is251_1 ... Is251_9	None	LS251 device. Example: "/IF=Is251_1".



### Example

Example for connection description:

CD="/IF=Is251\_1"

## 6.2.4 Configuration file in redundant operation

The PVI communication needs a unique station address for each station (PLC or computer) in the INA2000 network. For the zenon redundancy operation this means that the server and the standby need different configuration files. In the configuration file the station address is saved for each connection. This is how you can create different configuration files for the server and the standby:

- ▶ Set up the driver in the Editor so that the connection is correct.
- ▶ Test the communication at the server in the Runtime.
- ▶ Activate the standby once. At this the standby fetches the whole project including the configuration file from the server. Deactivate the standby.
- ▶ Open the **Remote Transport** configuration in the project properties.
- ▶ Select the line that is responsible for the driver files (line 8) and set this line to inactive. This setting makes sure that the changes you made at the standby are not overwritten by the file from the server. If you do not make this change, the standby will fetch the topical configuration file from the server and overwrite its own local file.
- ▶ Transfer this change to the server and reload it or start it again.
- ▶ Start the Windows Explorer at the standby.
- ▶ Go to the Runtime folder of the project and then to folder: \RT\FILES\zenon\custom\drivers. There you can find the driver configuration file. Open the file with the Windows Editor (Notepad).
- ▶ For each connection you can find entry DEVICE\_PARAM= in the file. In this entry the station address is saved (/SA=x). Change the station address to a unique station address. Save the changes and close the file.



- ▶ Create a variable in the Editor which has property **Read from Standby Server only** set. With the help of this variable you can check in the Runtime whether the server and the standby communicate simultaneously with the control.

**Attention**

*Changes in the driver configuration (e.g. changed IP address) in the Editor are now no longer transferred via the Remote Transport to the server and from there to the standby. You must manually take care that the changes are added to the configuration files at the server and the standby.*

## 7. Creating variables

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

### 7.1 Creating variables in the Editor

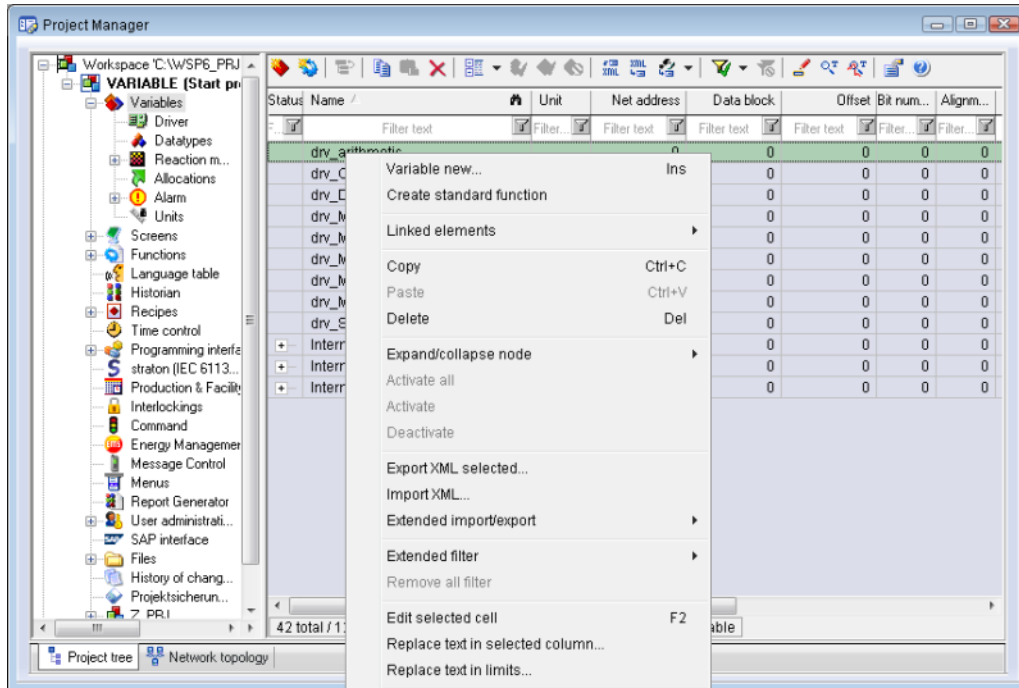
Variables can be created:

- ▶ 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:

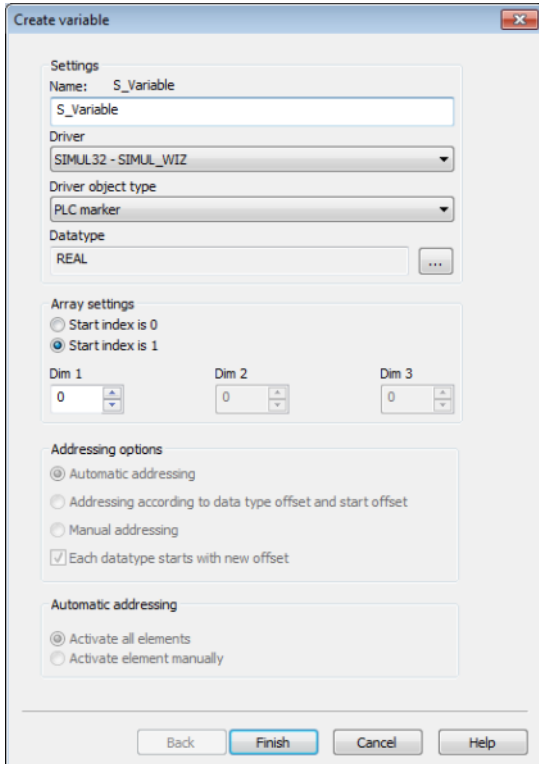
1. Select the **New variable** command in the **Variables** node in the context menu



The dialog for configuring variables is opened

2. Configure the variable

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



Property	Description
<b>Name</b>	<p>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.</p> <p>Maximum length: 128 characters</p> <p><b>Attention:</b> The characters # and @ are not permitted in variable names. If non-permitted characters are used, creation of variables cannot be completed and the <b>Finish</b> button remains inactive.</p> <p><b>Note:</b> For some drivers, the addressing is possible over the property <b>Symbolic address</b>, as well.</p>
<b>Drivers</b>	<p>Select the desired driver from the drop-down list.</p> <p><b>Note:</b> If no driver has been opened in the project, the driver for internal variables (<b>Intern.exe (Main.chm::/Intern.chm::/Intern.htm)</b>) is automatically loaded.</p>
<b>Driver Object Type</b> (cti.chm::/28685.htm)	<p>Select the appropriate driver object type from the drop-down list.</p>

<b>Data Type</b>	Select the desired data type. Click on the ... button to open the selection dialog.
<b>Array settings</b>	Expanded settings for array variables. You can find details in the Arrays chapter.
<b>Addressing options</b>	Expanded settings for arrays and structure variables. You can find details in the respective section.
<b>Automatic element activation</b>	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.

## 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
Addressing	Property group for addressing
Net address	Network address of variables. This address refers to the bus address in the connection configuration of the driver. This defines the PLC, on which the variable resides.
Data block	not used for this driver
Offset	Offset of variables. Equal to the memory address of the variable in the PLC. Adjustable from 0 to 4294967295.
Bit number	Number of the bit within the configured offset. Possible entries: 0 to 65535.
Alignment	not used for this driver
String length	Only available for String variables. Maximum number of characters that the variable can take.
PVI Address	Address information for the B&R PVI driver.
PVI Type (VT)	Variable type. (List of types see table: <b>PVI variable types</b> .)
PVI Number of Elements (VN)	Number of elements at field variables. Pre-allocation: VN=1. For multi-dimensional field variables the number of the elements of all field dimensions is quoted (Example: var[10][5] => VN =50).
PVI Length (VL)	Variable length in bytes. For single variables the variable length equals the process data length. For field variables the variable length defines the element length.
Driver connection	<i>Position of the variable within the defined offset.</i>
Drivers	Selection of the driver
Driver Object Type	Object type of the variables. Depending on the driver used, is selected when the variable is created and can be changed here.
Data Type	Data type of the variable. Is selected during the creation of the variable; the type can be changed here. <b>Attention:</b> If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.
PVI name	Is automatically set during online import.



### Attention

Addressing of one-dimensional arrays:

If one-dimensional rays are addressed with a start index  $\neq 0$ , a comma must be attached to the index. For details see: Documentation B&R.

For example: `zenon[3,]`

## PVI VARIABLE TYPE (VT)

Type	Description
i8	8 bit integer with sign. Variable length: VL=1. Range of values: -128 ... 127.
i16	16 bit integer with sign. Variable length: VL=2. Range of values: -32768 ... 32767.
i32	32 bit integer with sign. Variable length: VL=4. Range of values: -2147483648 ... 2147483647.
u8	8 bit integer unsigned. Variable length: VL=1. Range of values: 0 ... 255.
u16	16 bit integer unsigned. Variable length: VL=2. Range of values: 0 ... 65535.
u32	32 bit integer unsigned. Variable length: VL=4. Range of values: 0 ... 4294967295.
f32	32 bit floating point (IEEE floating). Variable length: VL=4. Range of values: -3.402823466e+38 ... -1.175494351e-38 / +1.175494351e-38 ... +3.402823466e+38.
f64	64 bit floating point (IEEE floating). Variable length: VL=8. Range of values: -1.7976931348623158e+308 ... -2.2250738585072014e-308 / +2.2250738585072014e-308 ... +1.7976931348623158e+308.
boolean	Bit variable (flag) mapped on 1 byte. Variable length: VL=1. TRUE = value not equal to 0, FALSE = value equal to 0.
string	String with 1 byte character size and binary 0 (zero) ending. The length of the string buffer can be defined via the variable length (parameter VL). The length of the string buffer is also the maximum string length. The actual string length defined by the binary zero character. At reading and writing process data with variable type string, take care that the data is only transferred up to and including the zero character. All characters after the zero character are undefined.

## 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	Description
<b>Reading in an interrogation</b>	36	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Read array polling as a whole. For details, see "Block reading of arrays" section.
<b>CPU status</b>	9	X	--	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	Status of CPU.
<b>PLC marker</b>	8	X	X	UINT, INT	Read data points polling/spontaneously.
<b>Communication details</b>	35	X	X	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING	<p>Variables for the static analysis of the communication; is transferred between driver and Runtime (not to the PLC).</p> <p><b>Note:</b> The addressing and the behavior is the same for most zenon drivers.</p> <p>You can find detailed information on this in the Communication details (Driver variables) (on page 65) chapter.</p>

#### Key:

**X:** supported

**--:** not supported

### BLOCK READING OF ARRAYS

If many elements of an array change very often, block reading of arrays can offer better performance than communication in the event of a value change. For this simple array variables and structures via driver object type **Read in an interrogation** can be read as block. A block array is a large variable in zenon that reads a complete block from the PLC. Many individual arrays are therefore read as a single array.

**Note:** If only a few elements change in an array, communication in Event Mode usually provides better performance.

The following applies for block reading:

- ▶ The complete array is always polled. Event-controlled communication is not possible.
- ▶ The priority selected must be the same for all elements.
- ▶ The PVI settings **VN**, **VL** and **VT** are only required for writing and ignored during reading. They must be set identically to **PLC marker**.
- ▶ Settings for import are defined in the Global Settings (on page 18) of the driver configuration.



#### Attention

*For block reading, zenon data type and PVI type must be the same.*

*This means: BOOL -> Boolean works. BOOL -> Boolean does not work.*

## EVENT OPERATION

**Attention:** The defined update times also have an effect on the communication with the control in Event mode.

This means: Events via value change never come more often than defined by the update time for the variable.

### 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.



Control	zenon	Data type
BOOL	BOOL	8
USINT	USINT	9
SINT	SINT	10
UINT	UINT	2
INT	INT	1
UDINT	UDINT	4
DINT	DINT	3
LREAL	LREAL	6
STRING	STRING	12

**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.



### 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

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.

## 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 cannot 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 (main.chm::/13046.htm)** 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 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 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.



### Information

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

## FILE STRUCTURE OF THE DBASE EXPORT FILE

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



### Attention

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

DBF files must:

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

## STRUCTURE

Identification	Type	Field size	Comment
KANALNAME	Char	128	Variable name.  The length can be limited using the <b>MAX_LAENGE</b> entry in the <b>project.ini</b> file.
KANAL_R	C	128	The original name of a variable that is to be replaced by the new name entered under "VARIABLENNAME" (variable name) (field/column must be entered manually).  The length can be limited using the <b>MAX_LAENGE</b> entry in the <b>project.ini</b> file.
KANAL_D	Log	1	The variable is deleted with the 1 entry (field/column has to be created by hand).
TAGNR	C	128	Identification.  The length can be limited using the <b>MAX_LAENGE</b> entry in the <b>project.ini</b> file.
EINHEIT	C	11	Technical unit
DATENART	C	3	Data type (e.g. bit, byte, word, ...) corresponds to the data type.
KANALTYP	C	3	Memory area in the PLC (e.g. marker area, data area, ...) corresponds to the driver object type.
HWKANAL	Num	3	Net address
BAUSTEIN	N	3	Datablock address (only for variables from the data area of the PLC)
ADRESSE	N	5	Offset
BITADR	N	2	For bit variables: bit address For byte variables: 0=lower, 8=higher byte For string variables: Length of string (max. 63 characters)
ARRAYSIZE	N	16	Number of variables in the array for index variables ATTENTION: Only the first variable is fully available. All others are only available for VBA or the Recipegroup Manager

<b>LES_SCHR</b>	L	1	Write-Read-Authorization 0: Not allowed to set value. 1: Allowed to set value.
<b>MIT_ZEIT</b>	R	1	time stamp in zenon (only if supported by the driver)
<b>OBJEKT</b>	N	2	Driver-specific ID number of the primitive object comprises TREIBER-OBJEKTYP and DATENTYP
<b>SIGMIN</b>	Float	16	Non-linearized signal - minimum (signal resolution)
<b>SIGMAX</b>	F	16	Non-linearized signal - maximum (signal resolution)
<b>ANZMIN</b>	F	16	Technical value - minimum (measuring range)
<b>ANZMAX</b>	F	16	Technical value - maximum (measuring range)
<b>ANZKOMMA</b>	N	1	Number of decimal places for the display of the values (measuring range)
<b>UPDATERATE</b>	F	19	Update rate for mathematics variables (in sec, one decimal possible) not used for all other variables
<b>MEMTIEFE</b>	N	7	Only for compatibility reasons
<b>HDRATE</b>	F	19	HD update rate for historical values (in sec, one decimal possible)
<b>HDTIEFE</b>	N	7	HD entry depth for historical values (number)
<b>NACHSORT</b>	R	1	HD data as postsorted values
<b>DRRATE</b>	F	19	Updating to the output (for zenon DDE server, in [s], one decimal possible)
<b>HYST_PLUS</b>	F	16	Positive hysteresis, from measuring range
<b>HYST_MINUS</b>	F	16	Negative hysteresis, from measuring range
<b>PRIOR</b>	N	16	Priority of the variable
<b>REAMATRIZE</b>	C	32	Allocated reaction matrix
<b>ERSATZWERT</b>	F	16	Substitute value, from measuring range
<b>SOLLMIN</b>	F	16	Minimum for set value actions, from measuring range
<b>SOLLMAX</b>	F	16	Maximum for set value actions, from measuring range
<b>VOMSTANDBY</b>	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
<b>RESOURCE</b>	C	128	Resources label. Free string for export and display in lists.  The length can be limited using the MAX_LAENGE entry in <b>project.ini</b> .
<b>ADJWVBA</b>	R	1	Non-linear value adaption: 0: Non-linear value adaption is used

			1: Non-linear value adaption is not used
<b>ADJZENON</b>	C	128	Linked VBA macro for reading the variable value for non-linear value adjustment.
<b>ADJWVBA</b>	C	128	ed VBA macro for writing the variable value for non-linear value adjustment.
<b>ZWREMA</b>	N	16	Linked counter REMA.
<b>MAXGRAD</b>	N	16	Gradient overflow for counter REMA.



### Attention

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

## LIMIT VALUE DEFINITION

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

Identification	Type	Field size	Comment
<b>AKTIV1</b>	R	1	Limit value active (per limit value available)
<b>GRENZWERT1</b>	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)
<b>SCHWWERT1</b>	F	16	Threshold value for limit value
<b>HYSTERESE1</b>	F	14	Is not used
<b>BLINKEN1</b>	R	1	Set blink attribute
<b>BTB1</b>	R	1	Logging in CEL
<b>ALARM1</b>	R	1	Alarm
<b>DRUCKEN1</b>	R	1	Printer output (for CEL or Alarm)
<b>QUITTIER1</b>	R	1	Must be acknowledged
<b>LOESCHE1</b>	R	1	Must be deleted
<b>VARIABLE1</b>	R	1	Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).
<b>FUNC1</b>	R	1	Functions linking
<b>ASK_FUNC1</b>	R	1	Execution via Alarm Message List
<b>FUNC_NR1</b>	N	10	ID number of the linked function (if "-1" is entered here, the existing function is not overwritten during import)
<b>A_GRUPPE1</b>	N	10	Alarm/Event Group
<b>A_KLASSE1</b>	N	10	Alarm/Event Class
<b>MIN_MAX1</b>	C	3	Minimum, Maximum
<b>FARBE1</b>	N	10	Color as Windows coding
<b>GRENZTXT1</b>	C	66	Limit value text
<b>A_DELAY1</b>	N	10	Time delay
<b>INVISIBLE1</b>	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.

### 7.4.3 Import PVI variables from the driver

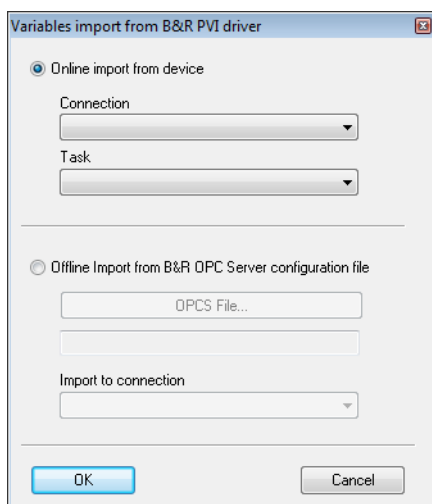
PVI variables can be imported offline via an OPCS XML file or via the online import of the driver.

At the import the variables are merged with existing ones. The key for this is the variable name.

- ▶ Variable designation
  - The variable name consists of: `Connection name.task name/variable name`
- ▶ Address information:
  - Address: `Task name/variable name`
  - Data type: `u16`
  - Connection allocation: Net address 0

To start the import:

- ▶ select **Import variables from driver...** from the context menu of the driver





Parameter	Description
<b>Offline Import from B &amp; R OPC server configuration file</b>	Import via XML file which the BuR OPC server generated.
<b>OPCS file...</b>	Selection of the OPCS XML file. Click on button in order to open the explorer for selecting the OPCS file.
<b>Import to connection</b>	Selection of the connection. A click opens the drop-down list with all available connections. All selected XML variables are connected during import to the selected connection via the net address
<b>Online import from device</b>	Variables are read online from the control.
<b>Connection</b>	Selection of the connection for the online browse.
<b>Task</b>	Selection of the task which is read out at the online browse.

## VARIABLE ADDRESSING

The addressing takes place via the properties of the applied PVI variable:

Parameter	Description
<b>Net address</b>	Determines the connection in the driver. Refers to the net address of the connection in the driver configuration dialog.
<b>PVI Name</b>	The name of the PLC variable. The name is created from Task name/variable name.
<b>PVI Type (VT)</b>	Variable type. A compilation of all variable types - see list below.
<b>PVI Number of Elements (VN)</b>	Number of elements at field variables. Pre-allocation: VN=1. For multi-dimensional field variables the number of the elements of all field dimensions is quoted (Example: var[10][5] => VN =50).
<b>PVI Length (VL)</b>	Variable length in bytes. For single variables the variable length equals the process data length. For field variables the variable length defines the element length. s

## GLOBAL VARIABLES

Global variables can only be used if they are integrated in a task. Global variables are therefore always displayed with the task variables during online import and also addressed as task variables: Task name/variable name.

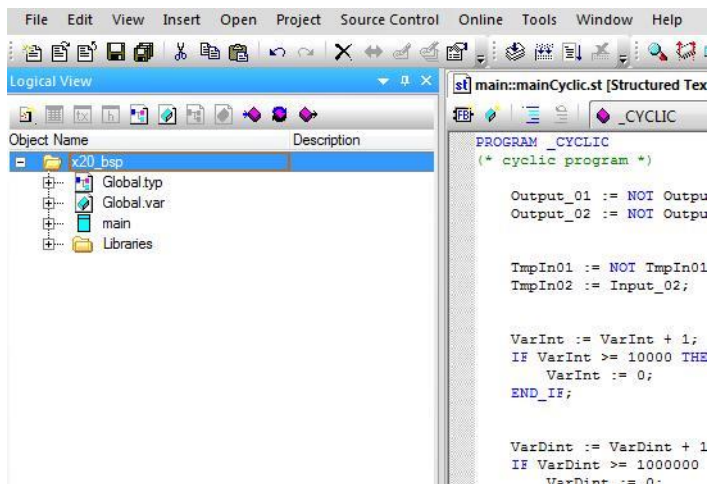
**PVI VARIABLE TYPE (VT)**

Type	Description
i8	8 bit integer with sign. Variable length: VL=1. Range of values: -128 ... 127.
i16	16 bit integer with sign. Variable length: VL=2. Range of values: -32768 ... 32767.
i32	32 bit integer with sign. Variable length: VL=4. Range of values: -2147483648 ... 2147483647.
u8	8 bit integer unsigned. Variable length: VL=1. Range of values: 0 ... 255.
u16	16 bit integer unsigned. Variable length: VL=2. Range of values: 0 ... 65535.
u32	32 bit integer unsigned. Variable length: VL=4. Range of values: 0 ... 4294967295.
f32	32 bit floating point (IEEE floating). Variable length: VL=4. Range of values: -3.402823466e+38 ... -1.175494351e-38 / +1.175494351e-38 ... +3.402823466e+38.
f64	64 bit floating point (IEEE floating). Variable length: VL=8. Range of values: -1.7976931348623158e+308 ... -2.2250738585072014e-308 / +2.2250738585072014e-308 ... +1.7976931348623158e+308.
boolean	Bit variable (flag) mapped on 1 byte. Variable length: VL=1. TRUE = value not equal to 0, FALSE = value equal to 0.
string	String with 1 byte character size and binary 0 (zero) ending. The length of the string buffer can be defined via the variable length (parameter VL). The length of the string buffer is also the maximum string length. The actual string length defined by the binary zero character. At reading and writing process data with variable type string, take care that the data is only transferred up to and including the zero character. All characters after the zero character are undefined.

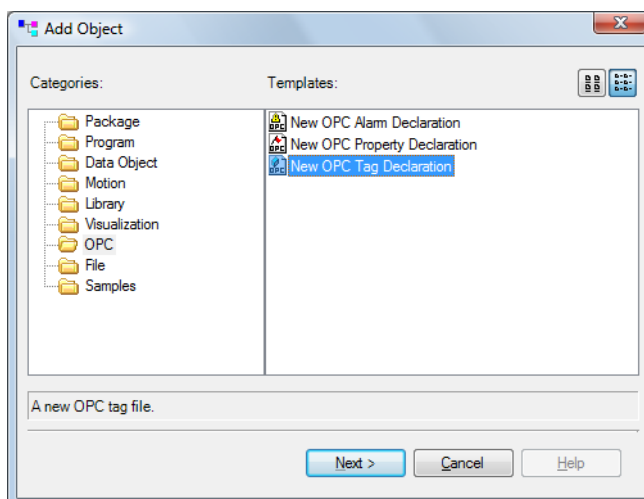
## Offline import

### PREPARATIONS IN THE AUTOMATION STUDIO

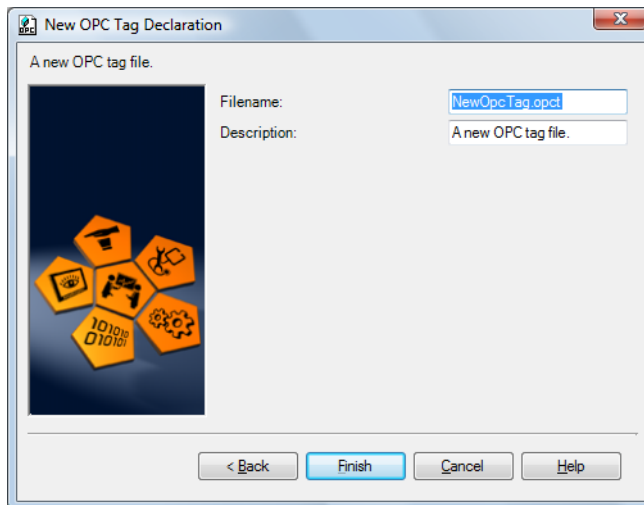
1. got to the desired file



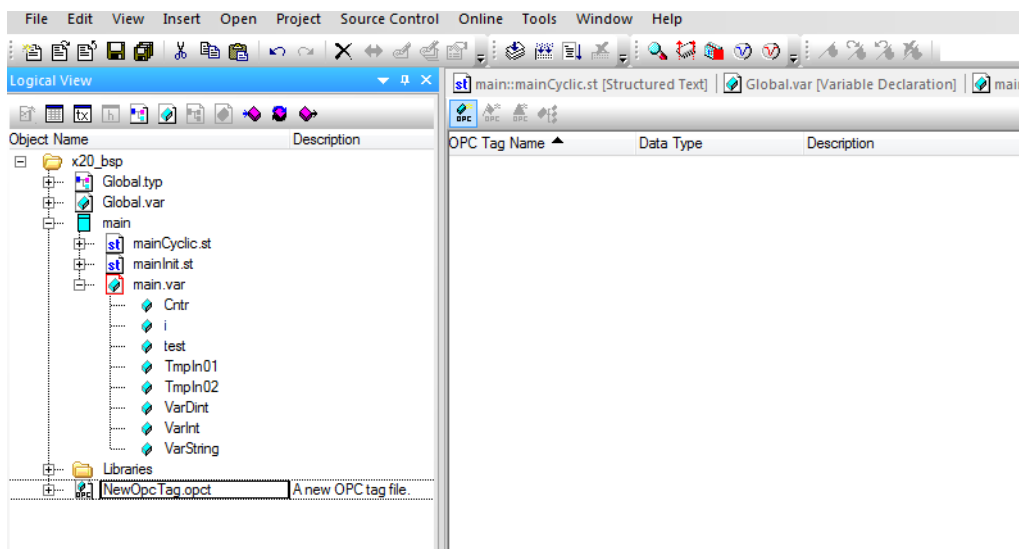
2. select Add Object... in the context menu.
3. in folder **OPC** select template **New OPC Tag Declaration**



4. assign a file name with extension .opct

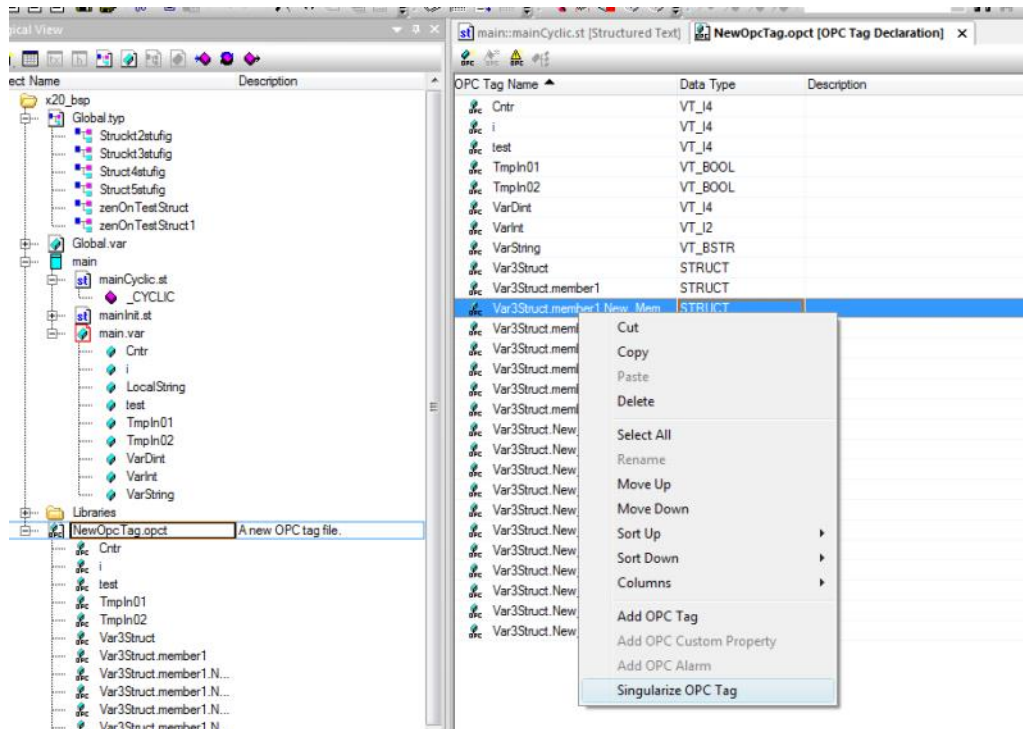


5. open the OPC tag list in Automation Studio

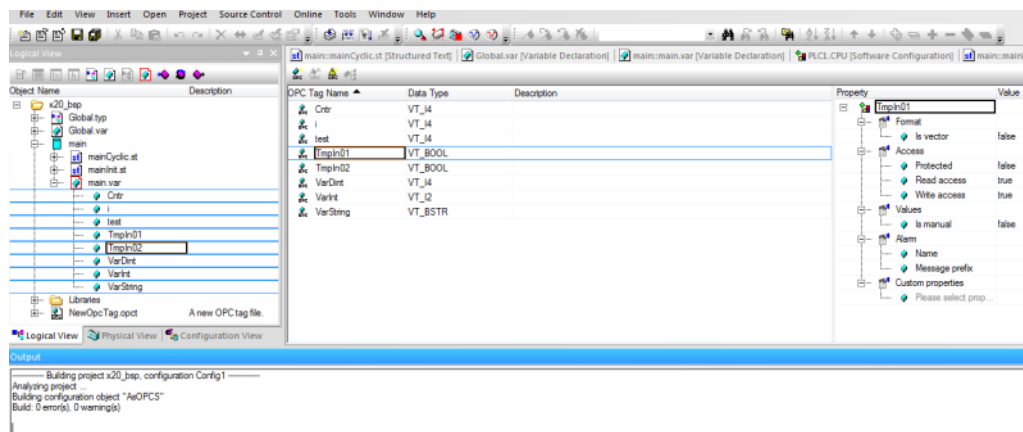


6. Add the variable you want to read out by means of drag & drop

- To be able to import structures and arrays offline too, the items of the structures must be removed.



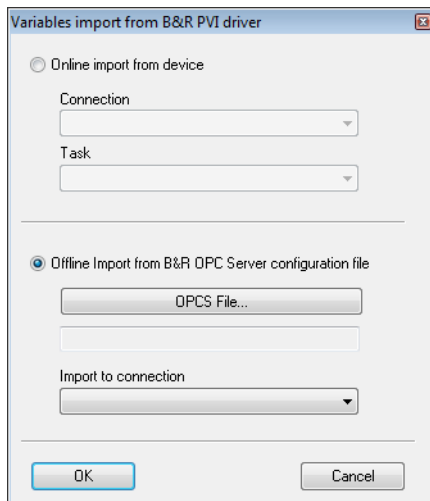
- at compiling the Automation Studio Project creates the OPCS file
- the OPCS file is stored in the project folder of the Automation Studio in a subfolder; e.g. `\Temp\Objects\Config1\PLC1\AsOPCS.opcs`



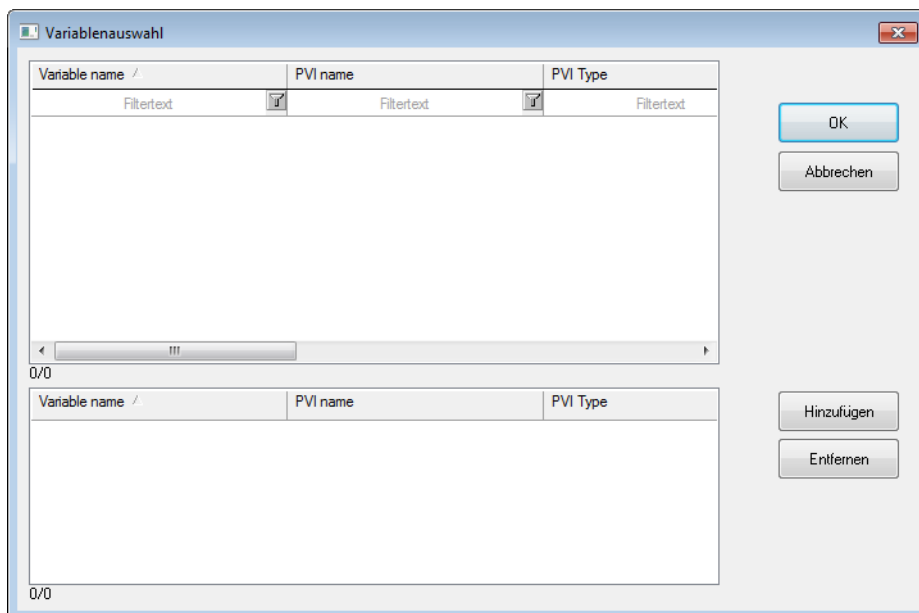
## IMPORT INTO ZENON

- Select the **Import variables from driver** command from the context menu of the driver
- in the import dialog select **Offline import from B&R OPC server configuration file**

3. click on **PPCS file...** and select the OPVS file.

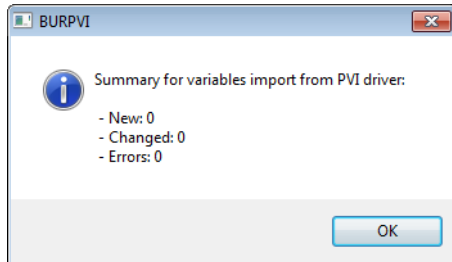


4. select the connection for which the import should be carried out
5. Confirm the dialog by clicking on OK
6. select the desired variable with the help of a double click or click on button **Add** (multi-selection is possible)



7. click on button OK in order to import the variable into zenon

8. an info box informs you about imported variables, changed variables and errors



## Online import

The driver also supports, for online import, multidimensional arrays and arrays with a start index  $\neq 0$  and creates variables with names that correspond to the original array index.



### Attention

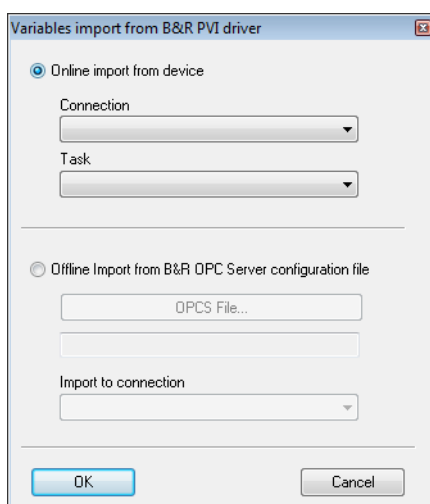
*Addressing of one-dimensional arrays:*

*If one-dimensional arrays are addressed with a start index  $\neq 0$ , a comma must be attached to the index. For details see: Documentation B&R.*

*For example: `zenon[3, ]`*

For the online import:

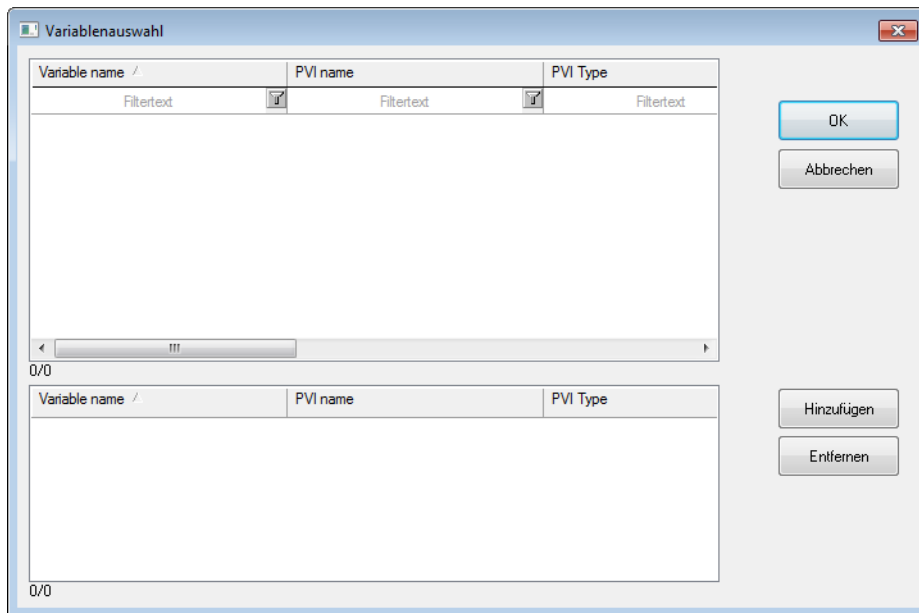
1. select command **Import variables from driver** from the context menu of the driver
2. select **Online Browse** in the import dialog
3. select the connection type and the desired task



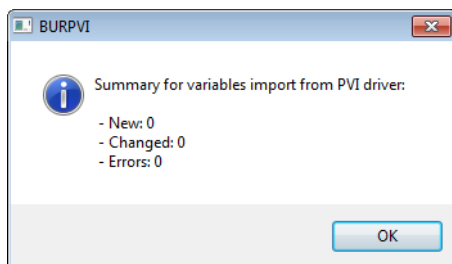
4. confirm the dialog by clicking OK



5. select the desired variable with the help of a double click or click on button **Add** (multi-selection is possible)



6. click on button OK in order to import the variable into zenon
7. an info box informs you about imported variables, changed variables and errors



## 7.5 Communication details (Driver variables)

The driver kit implements a number of driver variables. These 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** (on the installation medium in the \Predefined\Variables folder) and can be imported from there.

**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 only for pure polling drivers
- ▶ Connection-related information such as ErrorMSG only for drivers that only edit one connection at 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 terminated.
LineStateUnknow	BOOL	24.15	Connection status unknown
ModemStatus	UDINT	24	Current modem status
TreiberStop	BOOL	28	Driver stopped  For <code>driver stop</code> , the variable has the value <code>TRUE</code> and an <b>OFF</b> bit. After the driver has started, the variable has the value <code>FALSE</code> and no <b>OFF</b> bit.
SimulRTState	UDINT	60	Informs the status of Runtime for driver simulation.

ConnectionStates	STRING	61	<p>Internal connection status of the driver to the PLC.</p> <p>Connection statuses:</p> <p>0 : Connection OK</p> <p>1 : Connection failure</p> <p>2 : Connection simulated</p> <p>Formating:</p> <p>&lt;Netzadresse&gt;:&lt;Verbindungszustand&gt;;...;;</p> <p>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.</p> <p>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.</p>
------------------	--------	----	--

## 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 <b>PhoneNumberSet</b> and <b>ModemHwAdrSet</b> .
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).
RingCountSet	UDINT	20	Number of ringing tones before a call is accepted
ReCallIdleTime	UINT	53	Waiting time between calls in seconds (s).
ConnectTimeout	UINT	54	Time in seconds (s) to establish a connection.

## STATISTICS

Name from import	Type	Offset	Description
MaxWriteTime	UDINT	31	The longest time in milliseconds (ms) that is required for writing.
MinWriteTime	UDINT	32	The shortest time in milliseconds (ms) that is required for writing.
MaxBlkReadTime	UDINT	40	Longest time in milliseconds (ms) that is required to read a data block.
MinBlkReadTime	UDINT	41	Shortest time in milliseconds (ms) that is required to read a data block.
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 <b>Normal</b> in milliseconds (ms).
MaxUpdateTimeHigher	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).
MaxUpdateTimeHighest	UDINT	59	Time since the last update of the priority group <b>Highest</b> in milliseconds (ms).
PokeFinish	BOOL	55	Goes to 1 for a query, if all current pokes were executed

## ERROR MESSAGE

Name from import	Type	Offset	Description
------------------	------	--------	-------------

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

- ▶ Block arrays for structures and arrays
- ▶ For Online Import: Multi-dimensional arrays and arrays with a start index  $\neq 0$
- ▶ Blockwrite
- ▶ RDA

### RDA

A linear memory area in the control unit is needed for RDA archiving. This is implemented by means of an array. This array must contain the appropriate RDA header plus archive data. Only the Index [0] of

this array may be activated and marked as an RDA variable. If the array Index [0] is set to 1, the corresponding values are read out from the PLC.

**Note:** Variables configured with "**Only request from Standby Server**" are not supported in RDA archives.

You can find more information on RDA archiving in the zenon manual Archiving (Archivserver.chm::/28257.htm).

## 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

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



### Information

*This chapter describes standard functions that are valid for most zenon drivers. However, 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.*

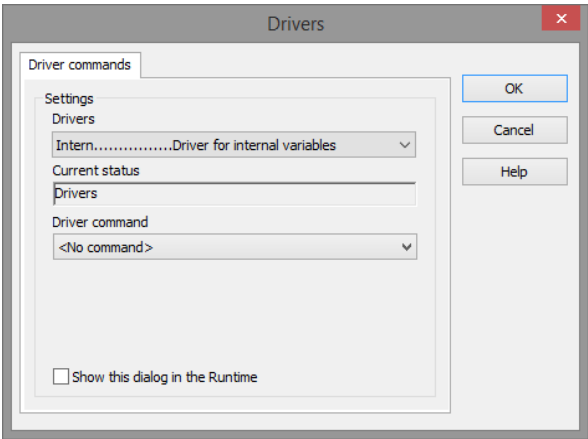
### CONFIGURATION OF THE FUNCTION

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

1. Create a new function in the zenon Editor.
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 Runtime. Heed the notices in the **Driver command function in the network** section.



**DRIVER COMMAND DIALOG**



Option	Description
<b>Drivers</b>	Selection of the driver from the drop-down list. It contains all drivers loaded in the project.
<b>Current status</b>	Fixed entry which has no function in the current version.
Driver command	Drop-down list for the selection of the command:
<No command>	No command is sent. A command that already exists can thus be removed from a configured function.
Start driver (online mode)	Driver is reinitialized 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 <code>switched off (OFF; Bit 20)</code> .
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	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)
Driver in counting simulation mode	Driver is set into counting simulation mode. All values are initialized with 0 and incremented in the set update time by 1 each time up to the maximum value and then start at 0 again.
Driver in static simulation mode	Driver is set into counting simulation mode. All values are initialized with 0.
Driver in programmed simulation mode	Driver is set into counting 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.
<b>Show this dialog in the Runtime</b>	The dialog is shown in Runtime so that changes can be made.

## 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, 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 8.00 -> Diagviewer*.

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

**%ProgramData%\COPA-DATA\LOG.**

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

In the Diagnosis Viewer you can also:

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

### Note:

1. The Diagnosis Viewer displays all entries in UTC (coordinated world time) and not in local time.
2. The Diagnosis Viewer does not display all columns of a LOG file per default. To display more columns activate property **Add all columns with entry** in the context menu of the column header.

3. If you only use **Error-Logging**, the problem description is in the column **Error text**. For other diagnosis level the description is in the column **General text**.
4. For communication problems many drivers also log error numbers which the PLC assigns to them. They are displayed in **Error text** or **Error code** or **Driver error parameter (1 and 2)**. Hints on the meaning of error codes can be found in the driver documentation and the protocol/PLC description.
5. At the end of your test set back the diagnosis level from **Debug** or **Deep Debug**. At **Debug** and **Deep Debug** there are a great deal of data for logging which are saved to the hard drive and which can influence your system performance. They are still logged even after you close the Diagnosis Viewer.



#### Attention

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

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

## 10.2 Check list

Checks after communication errors:

- ▶ Is the PLC connected to the power supply?
- ▶ Are the participants available in the **TCP/IP** network?
- ▶ Can the PLC be reached via the **Ping** command?
- ▶ Can the PLC be reached at the respective port via **TELNET**?
- ▶ Are the PLC and the PC connected with the right cable?
- ▶ Was the right **COM** port selected?
- ▶ Do the communication parameters match (Baud rate, parity, start/stop bits, ...)?
- ▶ Is the **COM** port blocked by another application?
- ▶ Did you configure the net address correctly, both in the driver dialog and in the address properties of the variables?
- ▶ Did you use the right object type for the variable?
- ▶ Does the offset addressing of the variable match the one in the PLC?
- ▶ Analysis with the Diagnosis Viewer: Which messages are displayed?

## 10.3 Error messages

Errors are documented in the output window or in the log file of the Diagnosis Viewer (main.chm::/12464.htm):

Entry	Debug Level	Meaning
<b>Block array item XY: Array index [Index] exceeds actual dimensions [Limit].</b>	ERROR	Given index (Address) overwrites the actual array limits.
<b>Block array item XY: Incompatible data types. (Data type size [Zahl] differs from actual data type size [Zahl])</b>	ERROR	zenon data types do not correspond to PVI type.
<b>Block item XY: Offset '%u' exceeds actual size of the object '%u'.</b>	ERROR	Given index (Address) overwrites the actual array/structure limits.
<b>Block item XY: Incompatible data types. (Data type size [Type1] differs from actual data type size [Type])</b>	ERROR	zenon data types do not correspond to PVI type.
<b>Block item XY: Struct element [Index] does not exist.</b>	ERROR	Indicated structure element (address) does not exist.