## Contents

1. Welcome to COPA-DATA help ........................................................................................................... 5
2. SAIA2ND32 ........................................................................................................................................ 5
3. SAIA2ND32 - Data sheet ...................................................................................................................... 6
4. Driver history ........................................................................................................................................ 7
5. Requirements ........................................................................................................................................ 8
   5.1 PC ................................................................................................................................................... 8
6. Configuration ......................................................................................................................................... 9
   6.1 Creating a driver .............................................................................................................................. 9
   6.2 Settings in the driver dialog .......................................................................................................... 11
       6.2.1 General ..................................................................................................................................... 11
       6.2.2 Driver dialog SAIA .................................................................................................................. 14
7. Creating variables ............................................................................................................................... 15
   7.1 Creating variables in the Editor .................................................................................................... 16
   7.2 Addressing ...................................................................................................................................... 19
   7.3 Driver objects and datatypes ........................................................................................................ 19
       7.3.1 Driver objects ........................................................................................................................ 20
       7.3.2 Mapping of the data types ..................................................................................................... 25
   7.4 Creating variables by importing .................................................................................................. 26
       7.4.1 XML import .......................................................................................................................... 27
       7.4.2 DBF Import/Export ............................................................................................................... 27
   7.5 Driver variables ............................................................................................................................. 32
8. Driver-specific functions ..................................................................................................................... 39
9. Driver commands .................................................................................................................................. 44
10. Error analysis ....................................................................................................................................... 46
    10.1 Analysis tool ............................................................................................................................... 46
    10.2 Error numbers .............................................................................................................................. 47
10.3 Check list
1. Welcome to COPA-DATA help

GENERAL HELP

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

PROJECT SUPPORT

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

LICENSES AND MODULES

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

2. SAIA2ND32

The SAIA2ND32 driver communicates with the SAIA-BURGESS controllers using the TCP/IP, SAIA S-Bus and SAIA P800 protocols.
# 3. SAIA2ND32 - Data sheet

<table>
<thead>
<tr>
<th>General:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver file name</td>
<td>SAIA2ND32.exe</td>
</tr>
<tr>
<td>Driver name</td>
<td>SAIA2ND32 driver</td>
</tr>
<tr>
<td>PLC manufacturer</td>
<td>Saia;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Driver supports:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>TCP/IP; SAIA S-Bus; SAIA P800;</td>
</tr>
<tr>
<td>Addressing: Address-based</td>
<td>X</td>
</tr>
<tr>
<td>Addressing: Name-based</td>
<td>--</td>
</tr>
<tr>
<td>Spontaneous communication</td>
<td>--</td>
</tr>
<tr>
<td>Polling communication</td>
<td>X</td>
</tr>
<tr>
<td>Online browsing</td>
<td>--</td>
</tr>
<tr>
<td>Offline browsing</td>
<td>--</td>
</tr>
<tr>
<td>Real-time capable</td>
<td>--</td>
</tr>
<tr>
<td>Blockwrite</td>
<td>--</td>
</tr>
<tr>
<td>Modem capable</td>
<td>X</td>
</tr>
<tr>
<td>Serial logging</td>
<td>--</td>
</tr>
<tr>
<td>RDA numerical</td>
<td>X</td>
</tr>
<tr>
<td>RDA String</td>
<td>--</td>
</tr>
</tbody>
</table>
4. **Driver history**

<table>
<thead>
<tr>
<th>Date</th>
<th>Driver version</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>07.07.08</td>
<td>1400</td>
<td>Created driver documentation</td>
</tr>
<tr>
<td>26.01.2016</td>
<td>7.60.0.25446</td>
<td>Support for [SAIA PG] software version PG5.2.x built in.</td>
</tr>
</tbody>
</table>

**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 form the next consecutive build number.
5. Requirements

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

5.1 PC

HARDWARE

There are the following hardware requirements for the use of the SAIA2ND32 driver:

- Connection:
  Serial interface RS232

- Cable:
  Converter cable

- Protocol:
  SAIAP 800/SAIA SBUS

SOFTWARE

The following files must be present in the zenon folder:

- Driver **SAIA2nd32.EXE**.

- The following files from Saia-Burgess
  (included in the installation setup of PG5 from SAIA):
  - SCommCom52.dll
  - SCommDlg52.dll
The driver requires these DLLs in the version 5.2 or newer.

**Note:** These files must not exist more than once on the PC!

If there already programs from **SAIA-BURGESS** that also contain these files installed on your computer, these must **NOT** be present in the zenon folder again.

Note this if you install the PG5 Suite on your computer, for example.

---

**Attention**

_from zenon 7.60, it is now possible to communicate with the controller using version 5.2 SAIA PG Software or higher._

_The support for older software versions has been discontinued by the manufacturer._

---

**CONNECTION**

The connection is established directly from the serial PC interface via a serial interface of the controller

---

### 6. Configuration

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

---

**Information**

_from the online manual:_

Find out more about further settings for zenon variables in the chapter **Variables** (main.chm::/15247.htm) of the online manual.

---

### 6.1 Creating a driver

In order to create a new driver:
1. Right-click on **Driver** in the Project Manage and select **Driver new** in the context menu.

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

3. Select the desired driver and give it a name:
   - The driver name has to be unique, i.e. if one and the same driver is to be used several times in one project, a new name has to be given each time.
   - The driver name is part of the file name. Therefore it may only contain characters which are supported by the operating system. Invalid characters are replaced by an underscore (_).
   - **Attention:** This name cannot be changed later on.
4. Confirm the dialog with **OK**. In the following dialog the single configurations of the drivers are defined.

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

**Information**

*For new projects and for existing projects which are converted to version 6.21 or higher, the following drivers are created automatically:*

- Internal
- MathDr32
- SysDrv.

**6.2 Settings in the driver dialog**

You can change the following settings of the driver:

**6.2.1 General**

The configuration dialog is opened when a driver is created. In order to be able to open the dialog later for editing, double click on the driver in the list or click on the **Configuration** property.
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Allows to switch between hardware mode and simulation mode</td>
</tr>
<tr>
<td></td>
<td>- Hardware:</td>
</tr>
<tr>
<td></td>
<td>A connection to the control is established.</td>
</tr>
<tr>
<td></td>
<td>- Simulation static</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>- Simulation - counting</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>- Simulation - programmed</td>
</tr>
<tr>
<td></td>
<td>N communication is established to the PLC. The values are calculated by a freely programmable simulation project. The simulation project is created with the help of the zenon Logic Workbench and runs in a zenon Logic Runtime which is integrated in the driver. For details see chapter Driver simulation (main.chm::/25206.htm).</td>
</tr>
<tr>
<td>Keep update list in the memory</td>
<td>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.</td>
</tr>
<tr>
<td>Output can be written</td>
<td>Active: Outputs can be written.</td>
</tr>
<tr>
<td></td>
<td>Inactive: Writing of outputs is prevented.</td>
</tr>
<tr>
<td></td>
<td>Note: Not available for every driver.</td>
</tr>
<tr>
<td>Variable image remanent</td>
<td>This option saves and restores the current value, time stamp and the states of a data point.</td>
</tr>
<tr>
<td></td>
<td>Fundamental requirement: The variable must have a valid value and time stamp.</td>
</tr>
</tbody>
</table>
The variable image is saved in mode hardware if:

- one of the states S_MERKER_1(0) up to S_MERKER8(7), REVISION(9), AUS(20) or ERSATZWERT(27) is active.

The variable image is always saved if:

- the variable is of the object type **Driver variable**
- the driver runs in simulation mode. (not programmed simulation)

The following states are not restored at the start of the Runtime:

- SELECT(8)
- WR-ACK(40)
- WR-SUC(41)

The mode **Simulation - programmed** at the driver start is not a criterion in order to restore the remanent variable image.

### Stop on Standby Server

Setting for redundancy at drivers which allow only one communication connection. For this the driver is stopped at the Standby Server and only started at the upgrade.

**Attention:** If this option is active, the gapless archiving is no longer guaranteed.

**Active:** Sets the driver at the not-process-leading Server automatically in a stop-like state. In contrast to stopping via driver command, the variable does not receive status switched off ([statusverarbeitung.chm::/24150.htm](statusverarbeitung.chm::/24150.htm)) but an empty value. This prevents that at the upgrade to the Server irrelevant values are created in the AML, CEL and Historian.

**Note:** Not available if the CE terminal serves as a data server. You can find further information in the zenon Operator manual in the CE terminal as a data server chapter.

### Global Update time

**Active:** The set Global update time in ms is used for all variables in the project. The priority set at the variables is not used.

**Inactive:** The set priorities are used for the individual variables.

### Priority

The polling times for the individual priority classes are set here. All variables with the according priority are polled in the set time.

The allocation to the variables takes place separately in the settings of the variable properties.

The communication of the individual variables are graduated in respect of importance or necessary topicality using the priorities.
Thus the communication load is distributed better.

Attention: Priority classes are not supported by each driver. For example, drivers that communicate spontaneously do not support it.

### CLOSE DIALOG

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OK</strong></td>
<td>Applies all changes in all tabs and closes the dialog.</td>
</tr>
<tr>
<td><strong>Cancel</strong></td>
<td>Discards all changes in all tabs and closes the dialog.</td>
</tr>
<tr>
<td><strong>Help</strong></td>
<td>Opens online help.</td>
</tr>
</tbody>
</table>

### UPDATE TIME FOR CYCLICAL DRIVERS

The following applies for cyclical drivers:

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

### 6.2.2 Driver dialog SAIA
Creating variables

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

Variables can be created:

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

VARIABLE DIALOG

To create a new variable, regardless of which type:

1. Select the **New variable** command in the **Variables** node in the context menu
2. The dialog for configuring variables is opened
3. Configure the variable
4. The settings that are possible depends on the type of variables
### Creating variables

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

### 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

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| Name                                    | Freely definable name
   *Attention:* the name must be unique within each control system project. |
| Identification                          | Any text can be entered here, e.g. for resource labels, comments ...        |
| Net address                             | Bus address or net address of the variable.                                 |
   *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                              | For variables of object type Extended data block, enter the datablock number here. |
   *Configurable [0.. 4294967295]. Please look up the exact maximum range for data blocks in the manual of the PLC.* |
| Offset                                  | Offset of the variable; the memory address of the variable in the PLC. Configurable [0.. 4294967295]. |
| Alignment                               | not used for this driver                                                   |
| Bit number                              | Number of the bit within the configured offset.                            |
   *Allowed entry [0.. 65535], Working range [0..7]* |
| String length                           | Only available for String variables: Maximum number of characters that the variable can take. |
| Driver connection/Driver Object Type    | Depending on the employed driver, an object type is selected during the creation of the variable; the type can be changed here later. |
| Driver connection/Data Type             | Data type of the variable, which is selected during the creation of the variable; the type can be changed here later. |
   *Attention: If you change the data type later, all other properties of the variable must be checked and adjusted, if necessary.* |
| Driver connection/Priority              | not used for this driver The driver does not support cyclically-poling communication in priority classes. |

### 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:
### Creating variables

<table>
<thead>
<tr>
<th>Driver object type</th>
<th>Channel type</th>
<th>Read</th>
<th>Write</th>
<th>Supported data types</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>11</td>
<td>X</td>
<td>X</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>Ext. Data block</td>
<td>34</td>
<td>X</td>
<td>X</td>
<td>UDINT, DINT, REAL, STRING, BOOL</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>10</td>
<td>X</td>
<td>--</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td>21</td>
<td>X</td>
<td>X</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>HW Status</td>
<td>64</td>
<td>X</td>
<td>--</td>
<td>UDINT, UINT</td>
<td></td>
</tr>
<tr>
<td>Register</td>
<td>8</td>
<td>X</td>
<td>X</td>
<td>UDINT, DINT, REAL, STRING, BOOL</td>
<td></td>
</tr>
<tr>
<td>Counter</td>
<td>23</td>
<td>X</td>
<td>X</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td>Timer</td>
<td>22</td>
<td>X</td>
<td>X</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td>Driver variable</td>
<td>35</td>
<td>X</td>
<td>X</td>
<td>BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING</td>
<td>Variables for the statistical analysis of communication. Find out more in the chapter about the Driver variables (on page 32)</td>
</tr>
</tbody>
</table>

### Objects for Process Variables in Zenon

<table>
<thead>
<tr>
<th>Object</th>
<th>Read</th>
<th>Write</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit-Register</td>
<td>X</td>
<td>X</td>
<td>bit one-step addressing (linear) Addressing: one-step via OFFSET</td>
</tr>
<tr>
<td>DWord-Register</td>
<td>X</td>
<td>X</td>
<td>32 bit double word one-step addressing (linear)</td>
</tr>
<tr>
<td>DataBit</td>
<td>X</td>
<td>X</td>
<td>Bit, is created in the data area of SAIA</td>
</tr>
<tr>
<td>data doubleword</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Input bit</td>
<td>X</td>
<td></td>
<td>Bit, directly accesses to the inputs Addressing: one-step via OFFSET</td>
</tr>
<tr>
<td>Output Bit</td>
<td>X</td>
<td></td>
<td>Bit, directly accesses to the outputs</td>
</tr>
</tbody>
</table>
Creating variables

<table>
<thead>
<tr>
<th></th>
<th>Addressing: one-step via OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timer</strong></td>
<td>X</td>
</tr>
<tr>
<td><strong>Counter</strong></td>
<td>X</td>
</tr>
<tr>
<td><strong>Flags</strong></td>
<td>X X</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>X</td>
</tr>
</tbody>
</table>

**Timer**
- 32 bit double word

**Counter**
- 32 bit double word

**Flags**
- Bit, directly accesses to the flags
- Addressing: one-step via OFFSET

**Status**
- 16 bit word

**PREDEFINED VARIABLES**

The variables, which the driver supports are defined in the Variable import file SAIA2nd32.DBF.

**VARIABLES OF TYPE STATUS:**

These are all data type WORD. It is recommended to use corresponding driver variables with zenon Version 5.2 and higher.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Offset</th>
<th>Write</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_ERROR</td>
<td>0</td>
<td>--</td>
<td>Is 65535 in case of error. Is 0 if there is not error.</td>
</tr>
<tr>
<td>ST_COMMTIMEOUT</td>
<td>1</td>
<td>--</td>
<td>IS 1 in case of interface timeout.</td>
</tr>
<tr>
<td>ST_ERRORCODE</td>
<td>2</td>
<td>--</td>
<td>Contains an error code according to SAIA and the driver errors.</td>
</tr>
<tr>
<td>ST_CSCOMMERROR</td>
<td>3</td>
<td>--</td>
<td>Is 1 in case of error. Is 0 if there is not error.</td>
</tr>
<tr>
<td>ST_ERRORCNT</td>
<td>4</td>
<td>X</td>
<td>Counts the failed transfers.</td>
</tr>
<tr>
<td>ST_CONNECT</td>
<td>5</td>
<td>X</td>
<td>If this is &lt;&gt; 0, then the connection will be established. If 0 the connection is cut.</td>
</tr>
<tr>
<td>ST_CONNECTDLG</td>
<td>6</td>
<td>X</td>
<td>Writing a value on this offset activates a dialog for connection settings.</td>
</tr>
<tr>
<td>ST_TELBOOK_NO</td>
<td>7</td>
<td>X</td>
<td>The content of this table element selects the telephone book entry. The 1st entry has the value 0. A value of 65535 denotes that no telephone book entry is to be used.</td>
</tr>
<tr>
<td>ST_SPSCLOCK</td>
<td>8</td>
<td>X</td>
<td>Is of type Status long. The number of seconds since 1.1. 1970 0:0</td>
</tr>
</tbody>
</table>

**DRIVER VARIABLES**

The driver supports the following driver variables:
Creating variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Offset</th>
<th>Write</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaiaERROR</td>
<td>Bit</td>
<td>1000</td>
<td>--</td>
<td>Is 1 in case of error.</td>
</tr>
<tr>
<td>SaiaCOMMTIMEOUT</td>
<td>Bit</td>
<td>1001</td>
<td>--</td>
<td>IS 1 in case of interface timeout.</td>
</tr>
<tr>
<td>SaiaErrorCode</td>
<td>DWORD</td>
<td>1002</td>
<td>--</td>
<td>The error code of driver and PCD-Connection. The error codes are moved to offset 128. In order to prevent an overlapping of internal error codes.</td>
</tr>
<tr>
<td>SaiaCSCOMMERROR</td>
<td>Bit</td>
<td>1003</td>
<td>--</td>
<td>IS 1 in case of interface error.</td>
</tr>
<tr>
<td>SaiaERRORCNT</td>
<td>DWORD</td>
<td>1004</td>
<td>X</td>
<td>Number of errors.</td>
</tr>
<tr>
<td>SaiaVERBIND</td>
<td>Bit</td>
<td>1005</td>
<td>X</td>
<td>If 1 the connection is established.</td>
</tr>
<tr>
<td>SaiaCONNECTDLG</td>
<td>Bit</td>
<td>1006</td>
<td>X</td>
<td>Writing on this address activates the connection dialog.</td>
</tr>
<tr>
<td>SaiaTELBOOK_NO</td>
<td>DWORD</td>
<td>1007</td>
<td>X</td>
<td>Selects the telephone book entry.</td>
</tr>
<tr>
<td>SaiaConnected</td>
<td>Bit</td>
<td>1008</td>
<td>--</td>
<td>1 if a connection exists.</td>
</tr>
<tr>
<td>SaiaPhoneNo</td>
<td>String</td>
<td>1009</td>
<td>X</td>
<td>Telephone number for selection.</td>
</tr>
<tr>
<td>SaiaBusNo</td>
<td>DWORD</td>
<td>1010</td>
<td>X</td>
<td>Bus number for selection.</td>
</tr>
<tr>
<td>SaiaApply</td>
<td>Bit</td>
<td>1011</td>
<td>X</td>
<td>If 1 the done settings of variable SaiaPhoneNo and SaiaBusNo are activated. This variable is automatically set back to 0.</td>
</tr>
<tr>
<td>SaiaConnectedStation</td>
<td>DWORD</td>
<td>1014</td>
<td>--</td>
<td>Contains the S-Bus address of the PLC that the connection was established with.</td>
</tr>
<tr>
<td>SaiaOnOff</td>
<td>STRING</td>
<td>1015</td>
<td>X</td>
<td>Specifies the S-Bus addresses that are not read. Data points with this net address will not be modified and are therefore frozen. If you enter several addresses, you must separate them with a semicolon (;). The value ALL locks these addresses.</td>
</tr>
</tbody>
</table>

For this driver variable, an import description is defined in the file Saia2nd32.dbf.

**DRIVER COMMANDS:**

The driver supports the driver commands "Connect driver modem" and "Hang up driver modem". These commands are not restricted to one special modem connection but also establish a connection to S-bus and PGU.
Creating variables

For S-Bus Connection:

Net Address:

The used net address is put together as follows:

If the net address of a variable is set to another value than 0, this address will be used.

If in the driver variable SaiaBusno has a value that is not 0, this value is used. The content of this variable is changed by the "Connect driver modem" driver command.

If a telephone book entry is selected and the station number is unequal 0, this setting is used by the telephone book.

For PGU Connection:

Net Address = 0

The objects consist of data from the SAIA PLC that are usually created in zenon. The properties of the objects are defined by the used PLC.

Statusobject

With the statusobject with address 5 it is possible to select via zenon. If the value 1 is written to a status object, the connection is established, write 0 to the status word to cancel the modem connection.

Information

Also with normal serial connection the status object with the address 5 must be set to 1. This results in the establishment of the connection from zenon to the Saia PLC.

If the status is not set to 1, no connection is established.

Modem plug by SAIA PCD2 to 9-pin. Sub. D Connector

Exeption: Error counter can be set back only via set value.

Also with normal serial connection the status object with the address 5 must be set to 1. This causes the connection between zenon and the SAIA PLC.

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.
### Creating variables

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.*

<table>
<thead>
<tr>
<th>PLC</th>
<th>zenon</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>USINT</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>SINT</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>UINT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>UDINT</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DINT</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>UUINT</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>LINT</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>LREAL</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>WSTRING</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>DATE_AND_TIME</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>TOD (Time of Day)</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

**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.
7.4.1 XML import

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

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

⚠️ Attention

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

*Example:*

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

7.4.2 DBF Import/Export

Data can be exported to and imported from dBase.

💡 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
Creating variables

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)

<table>
<thead>
<tr>
<th>STRUCTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identification</strong></td>
</tr>
<tr>
<td>KANALNAME</td>
</tr>
<tr>
<td>KANAL_R</td>
</tr>
<tr>
<td>KANAL_D</td>
</tr>
<tr>
<td>TAGNR</td>
</tr>
<tr>
<td>EINHEIT</td>
</tr>
<tr>
<td>DATENART</td>
</tr>
<tr>
<td>KANALTYP</td>
</tr>
<tr>
<td>HWKANAL</td>
</tr>
<tr>
<td>BAUSTEIN</td>
</tr>
<tr>
<td>ADRESSE</td>
</tr>
<tr>
<td>BITADR</td>
</tr>
<tr>
<td>ARRAYSIZE</td>
</tr>
<tr>
<td>Variable</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>LES_SCHR</td>
</tr>
<tr>
<td>MIT_ZEIT</td>
</tr>
<tr>
<td>OBJEKT</td>
</tr>
<tr>
<td>SIGMIN</td>
</tr>
<tr>
<td>SIGMAX</td>
</tr>
<tr>
<td>ANZMIN</td>
</tr>
<tr>
<td>ANZMAX</td>
</tr>
<tr>
<td>ANZKOMMA</td>
</tr>
<tr>
<td>UPDATERATE</td>
</tr>
<tr>
<td>MEMTIEFE</td>
</tr>
<tr>
<td>HDRATE</td>
</tr>
<tr>
<td>HDTIEFE</td>
</tr>
<tr>
<td>NACHSORT</td>
</tr>
<tr>
<td>DRRATE</td>
</tr>
<tr>
<td>HYST_PLUS</td>
</tr>
<tr>
<td>HYST_MINUS</td>
</tr>
<tr>
<td>PRIOR</td>
</tr>
<tr>
<td>REAMATRIZE</td>
</tr>
<tr>
<td>ERSATZWERT</td>
</tr>
<tr>
<td>SOLLMIN</td>
</tr>
<tr>
<td>SOLLMAX</td>
</tr>
<tr>
<td>VOMSTANDBY</td>
</tr>
<tr>
<td>RESOURCE</td>
</tr>
<tr>
<td>ADJWVBA</td>
</tr>
</tbody>
</table>
Creating variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJZENON</td>
<td>C</td>
<td>128</td>
<td>Linked VBA macro for reading the variable value for non-linear value adjustment.</td>
</tr>
<tr>
<td>ADJWVBA</td>
<td>C</td>
<td>128</td>
<td>ed VBA macro for writing the variable value for non-linear value adjustment.</td>
</tr>
<tr>
<td>ZWREMA</td>
<td>N</td>
<td>16</td>
<td>Linked counter REMA.</td>
</tr>
<tr>
<td>MAXGRAD</td>
<td>N</td>
<td>16</td>
<td>Gradient overflow for counter REMA.</td>
</tr>
</tbody>
</table>

**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:
<table>
<thead>
<tr>
<th>Identification</th>
<th>Type</th>
<th>Field size</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKTIV1</td>
<td>L</td>
<td>1</td>
<td>Limit value active (per limit value available)</td>
</tr>
<tr>
<td>GRENZWERT1</td>
<td>F</td>
<td>20</td>
<td>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)</td>
</tr>
<tr>
<td>SCHWWERT1</td>
<td>F</td>
<td>16</td>
<td>Threshold value for limit value</td>
</tr>
<tr>
<td>HYSTERESE1</td>
<td>F</td>
<td>14</td>
<td>Is not used</td>
</tr>
<tr>
<td>BLINKEN1</td>
<td>L</td>
<td>1</td>
<td>Set blink attribute</td>
</tr>
<tr>
<td>BTB1</td>
<td>L</td>
<td>1</td>
<td>Logging in CEL</td>
</tr>
<tr>
<td>ALARM1</td>
<td>L</td>
<td>1</td>
<td>Alarm</td>
</tr>
<tr>
<td>DRUCKEN1</td>
<td>L</td>
<td>1</td>
<td>Printer output (for CEL or Alarm)</td>
</tr>
<tr>
<td>QUITTIER1</td>
<td>L</td>
<td>1</td>
<td>Must be acknowledged</td>
</tr>
<tr>
<td>LOESCHE1</td>
<td>L</td>
<td>1</td>
<td>Must be deleted</td>
</tr>
<tr>
<td>VARIABLE1</td>
<td>L</td>
<td>1</td>
<td>Dyn. limit value linking the limit is defined by an absolute value (see field GRENZWERTx).</td>
</tr>
<tr>
<td>FUNC1</td>
<td>L</td>
<td>1</td>
<td>Functions linking</td>
</tr>
<tr>
<td>ASK_FUNC1</td>
<td>L</td>
<td>1</td>
<td>Execution via Alarm Message List</td>
</tr>
<tr>
<td>FUNC_NR1</td>
<td>N</td>
<td>10</td>
<td>ID number of the linked function (if “-1” is entered here, the existing function is not overwritten during import)</td>
</tr>
<tr>
<td>A_GRUPPE1</td>
<td>N</td>
<td>10</td>
<td>Alarm/Event Group</td>
</tr>
<tr>
<td>A_KLASSE1</td>
<td>N</td>
<td>10</td>
<td>Alarm/Event Class</td>
</tr>
<tr>
<td>MIN_MAX1</td>
<td>C</td>
<td>3</td>
<td>Minimum, Maximum</td>
</tr>
<tr>
<td>FARBEI</td>
<td>N</td>
<td>10</td>
<td>Color as Windows coding</td>
</tr>
<tr>
<td>GRENZTXT1</td>
<td>C</td>
<td>66</td>
<td>Limit value text</td>
</tr>
<tr>
<td>A_DELAY1</td>
<td>N</td>
<td>10</td>
<td>Time delay</td>
</tr>
<tr>
<td>INVISIBLE1</td>
<td>L</td>
<td>1</td>
<td>Invisible</td>
</tr>
</tbody>
</table>

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

### 7.5 Driver variables

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

- 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 are to be imported from `drvvar.dbf` again, the variables that were imported beforehand must be renamed.
**Information**

Not every driver supports all driver variants.

For example:

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

<table>
<thead>
<tr>
<th>Name from import</th>
<th>Type</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MainVersion</td>
<td>UINT</td>
<td>0</td>
<td>Main version number of the driver.</td>
</tr>
<tr>
<td>SubVersion</td>
<td>UINT</td>
<td>1</td>
<td>Sub version number of the driver.</td>
</tr>
<tr>
<td>BuildVersion</td>
<td>UINT</td>
<td>29</td>
<td>Build version number of the driver.</td>
</tr>
<tr>
<td>RTMajor</td>
<td>UINT</td>
<td>49</td>
<td>zenon main version number</td>
</tr>
<tr>
<td>RTMinor</td>
<td>UINT</td>
<td>50</td>
<td>zenon sub version number</td>
</tr>
<tr>
<td>RTSp</td>
<td>UINT</td>
<td>51</td>
<td>zenon Service Pack number</td>
</tr>
<tr>
<td>RTBuild</td>
<td>UINT</td>
<td>52</td>
<td>zenon build number</td>
</tr>
<tr>
<td>LineStateIdle</td>
<td>BOOL</td>
<td>24.0</td>
<td>TRUE, if the modem connection is idle</td>
</tr>
<tr>
<td>LineStateOffering</td>
<td>BOOL</td>
<td>24.1</td>
<td>TRUE, if a call is received</td>
</tr>
<tr>
<td>LineStateAccepted</td>
<td>BOOL</td>
<td>24.2</td>
<td>The call is accepted</td>
</tr>
<tr>
<td>LineStateDialtone</td>
<td>BOOL</td>
<td>24.3</td>
<td>Dialtone recognized</td>
</tr>
<tr>
<td>LineStateDialing</td>
<td>BOOL</td>
<td>24.4</td>
<td>Dialing active</td>
</tr>
<tr>
<td>LineStateRingBack</td>
<td>BOOL</td>
<td>24.5</td>
<td>While establishing the connection</td>
</tr>
<tr>
<td>LineStateBusy</td>
<td>BOOL</td>
<td>24.6</td>
<td>Target station is busy</td>
</tr>
</tbody>
</table>
Creating variables

<table>
<thead>
<tr>
<th>Name from import</th>
<th>Type</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReconnectInRead</td>
<td>BOOL</td>
<td>27</td>
<td>If TRUE, the modem is automatically reconnected for reading</td>
</tr>
<tr>
<td>ApplyCom</td>
<td>BOOL</td>
<td>36</td>
<td>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).</td>
</tr>
<tr>
<td>ApplyModem</td>
<td>BOOL</td>
<td>37</td>
<td>Apply changes in the settings of the modem. Writing this variable immediately calls the method SrvDrvVarApplyModem. This closes the current connection and opens a new one according to the settings PhoneNumberSet and ModemHwAdrSet.</td>
</tr>
</tbody>
</table>

**CONFIGURATION**

<table>
<thead>
<tr>
<th>Name from import</th>
<th>Type</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TreiberStop</td>
<td>BOOL</td>
<td>28</td>
<td>Driver stopped For driver stop, the variable has the value TRUE and an OFF bit. After the driver has started, the variable has the value FALSE and no OFF bit.</td>
</tr>
<tr>
<td>SimulRTState</td>
<td>UDINT</td>
<td>60</td>
<td>Informs the status of Runtime for driver simulation.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Data Type</td>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PhoneNumberSet</td>
<td>STRING</td>
<td>38</td>
<td>Telephone number, that should be used</td>
</tr>
<tr>
<td>ModemHwAdrSet</td>
<td>DINT</td>
<td>39</td>
<td>Hardware address for the telephone number</td>
</tr>
<tr>
<td>GlobalUpdate</td>
<td>UDINT</td>
<td>3</td>
<td>Update time in milliseconds (ms).</td>
</tr>
<tr>
<td>BGlobalUpdaten</td>
<td>BOOL</td>
<td>4</td>
<td>TRUE, if update time is global</td>
</tr>
<tr>
<td>TreiberSimul</td>
<td>BOOL</td>
<td>5</td>
<td>TRUE, if driver in sin simulation mode</td>
</tr>
<tr>
<td>TreiberProzab</td>
<td>BOOL</td>
<td>6</td>
<td>TRUE, if the variables update list should be kept in the memory</td>
</tr>
<tr>
<td>ModemActive</td>
<td>BOOL</td>
<td>7</td>
<td>TRUE, if the modem is active for the driver</td>
</tr>
<tr>
<td>Device</td>
<td>STRING</td>
<td>8</td>
<td>Name of the serial interface or name of the modem</td>
</tr>
<tr>
<td>ComPort</td>
<td>UINT</td>
<td>9</td>
<td>Number of the serial interface.</td>
</tr>
<tr>
<td>Baudrate</td>
<td>UDINT</td>
<td>10</td>
<td>Baud rate of the serial interface.</td>
</tr>
<tr>
<td>Parity</td>
<td>SINT</td>
<td>11</td>
<td>Parity of the serial interface.</td>
</tr>
<tr>
<td>ByteSize</td>
<td>USINT</td>
<td>14</td>
<td>Number of bits per character of the serial interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Value = 0 if the driver cannot establish any serial connection.</td>
</tr>
<tr>
<td>StopBit</td>
<td>USINT</td>
<td>13</td>
<td>Number of stop bits of the serial interface.</td>
</tr>
<tr>
<td>Autoconnect</td>
<td>BOOL</td>
<td>16</td>
<td>TRUE, if the modem connection should be established automatically for reading/writing</td>
</tr>
<tr>
<td>PhoneNumber</td>
<td>STRING</td>
<td>17</td>
<td>Current telephone number</td>
</tr>
<tr>
<td>ModemHwAdr</td>
<td>DINT</td>
<td>21</td>
<td>Hardware address of current telephone number</td>
</tr>
<tr>
<td>RxIdleTime</td>
<td>UINT</td>
<td>18</td>
<td>Modem is disconnected, if no data transfer occurs for this time in seconds (s)</td>
</tr>
</tbody>
</table>
### Creating variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WriteTimeout</td>
<td>UDINT</td>
<td>19</td>
<td>Maximum write duration for a modem connection in milliseconds (ms).</td>
</tr>
<tr>
<td>RingCountSet</td>
<td>UDINT</td>
<td>20</td>
<td>Number of ringing tones before a call is accepted</td>
</tr>
<tr>
<td>ReCallIdleTime</td>
<td>UINT</td>
<td>53</td>
<td>Waiting time between calls in seconds (s).</td>
</tr>
<tr>
<td>ConnectTimeout</td>
<td>UINT</td>
<td>54</td>
<td>Time in seconds (s) to establish a connection.</td>
</tr>
</tbody>
</table>

### STATISTICS

<table>
<thead>
<tr>
<th>Name from import</th>
<th>Type</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxWriteTime</td>
<td>UDINT</td>
<td>31</td>
<td>The longest time in milliseconds (ms) that is required for writing.</td>
</tr>
<tr>
<td>MinWriteTime</td>
<td>UDINT</td>
<td>32</td>
<td>The shortest time in milliseconds (ms) that is required for writing.</td>
</tr>
<tr>
<td>MaxBlkReadTime</td>
<td>UDINT</td>
<td>40</td>
<td>Longest time in milliseconds (ms) that is required to read a data block.</td>
</tr>
<tr>
<td>MinBlkReadTime</td>
<td>UDINT</td>
<td>41</td>
<td>Shortest time in milliseconds (ms) that is required to read a data block.</td>
</tr>
<tr>
<td>WriteErrorCount</td>
<td>UDINT</td>
<td>33</td>
<td>Number of writing errors</td>
</tr>
<tr>
<td>ReadSucceedCount</td>
<td>UDINT</td>
<td>35</td>
<td>Number of successful reading attempts</td>
</tr>
</tbody>
</table>
### Creating variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxCycleTime</td>
<td>UDINT</td>
<td>22</td>
<td>Longest time in milliseconds (ms) required to read all requested data.</td>
</tr>
<tr>
<td>MinCycleTime</td>
<td>UDINT</td>
<td>23</td>
<td>Shortest time in milliseconds (ms) required to read all requested data.</td>
</tr>
<tr>
<td>WriteCount</td>
<td>UDINT</td>
<td>26</td>
<td>Number of writing attempts</td>
</tr>
<tr>
<td>ReadErrorCount</td>
<td>UDINT</td>
<td>34</td>
<td>Number of reading errors</td>
</tr>
<tr>
<td>MaxUpdateTimeNormal</td>
<td>UDINT</td>
<td>56</td>
<td>Time since the last update of the priority group Normal in milliseconds (ms).</td>
</tr>
<tr>
<td>MaxUpdateTimeHigher</td>
<td>UDINT</td>
<td>57</td>
<td>Time since the last update of the priority group Higher in milliseconds (ms).</td>
</tr>
<tr>
<td>MaxUpdateTimeHigh</td>
<td>UDINT</td>
<td>58</td>
<td>Time since the last update of the priority group High in milliseconds (ms).</td>
</tr>
<tr>
<td>MaxUpdateTimeHighest</td>
<td>UDINT</td>
<td>59</td>
<td>Time since the last update of the priority group Highest in milliseconds (ms).</td>
</tr>
<tr>
<td>PokeFinish</td>
<td>BOOL</td>
<td>55</td>
<td>Goes to 1 for a query, if all current pokes were executed</td>
</tr>
</tbody>
</table>

### ERROR MESSAGE

<table>
<thead>
<tr>
<th>Name from import</th>
<th>Type</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorTimeDW</td>
<td>UDINT</td>
<td>2</td>
<td>Time (in seconds since 1.1.1970), when the last error occurred.</td>
</tr>
<tr>
<td>ErrorTimeS</td>
<td>STRING</td>
<td>2</td>
<td>Time (in seconds since 1.1.1970), when the last error occurred.</td>
</tr>
<tr>
<td>RdErrPrimObj</td>
<td>UDINT</td>
<td>42</td>
<td>Number of the PrimObject, when the last reading error occurred.</td>
</tr>
<tr>
<td>RdErrStationsName</td>
<td>STRING</td>
<td>43</td>
<td>Name of the station, when the last reading error occurred.</td>
</tr>
<tr>
<td>RdErrBlockCount</td>
<td>UINT</td>
<td>44</td>
<td>Number of blocks to read when the last reading error occurred.</td>
</tr>
</tbody>
</table>
8. Driver-specific functions

The driver supports the following functions:

MODEM COMMUNICATION

HARDWARE

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCD1 M1xx</td>
<td>Function module interface RS 232 (Modem)  PCD7 F120 for channel 1</td>
</tr>
<tr>
<td>PCD2 M1xx</td>
<td>Function module interface RS 232 (Modem)  PCD7 F120 for channel 1</td>
</tr>
</tbody>
</table>

This PLC only offers channel 1 for modem communication The connection of the modem is explained in the according manuals.
SOFTWARE

COMMUNICATION SAIA PCD TO ZENON

Initialization of interface
The illustration shows an example program, in which a communication from SAIA PCD to zenon can be established. The settings concerning initialization of interface (always canal 1) are done in „SASI Modem“ data block. The „SASI Diag“ component provides a diagnosis for possible errors. In this component only the canal is selected. The single settings and assignment of Fupla-components are explained in the Online Help and SAIA-handbooks. The most important settings of the „SASI Moden“ component are shown in the following figure.

The settings of the Modem depends on the Modem manufacturer. The settings are saved in „modmsms.def“ in the folder *\SAIA\FBox\* How the settings have to be done is explained in the SAIA handbooks.
CALL-MASTER COMPONENT

The illustration shows another example program, in which a communication from SAIA PCD to zenon can be established. In the „Call Master“ Component the selection settings are done.

The single settings and assignment of Fupla-components are explained in the Online Help and SAIA-handbooks.

The most important settings of the „Call Master“ component are shown in the following figure.
COMMUNICATION ZENON TO SAIA PCD

The SAIA PCE is configured in the "PCD Configurator". In the following figure the necessary steps are described.

In the „Project manager“ under „Tools“ the „PCD Configurator“ is started.

The settings for the modem-communication are done in this „PCD Configurator“. Notice: Settings can only be changed, if an online connection to PCD exists.

Click „S-BUS“ and the dialog for „S-Bus“-definition opens.
In this dialog, the settings are done analogically. Configuration of the modem type depends on the manufacturer. Click on the Button „Modem“ to insert the Modem string. To access further S-Bus- Stations, activate the “Gateway”. The exact configuration is described in the SAIA handbooks. The settings are stored in the „Spg4modm.ini“ file. This file must be stored in the Windows directory.

9. Driver commands

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

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

- create a new function
- select Variables -> Driver commands
- The dialog for configuration is opened
## Parameter | Description
--- | ---
**Drivers** | Drop-down list with all drivers which are loaded in the project.
**Current status** | Fixed entry which has no function in the current version.
**Driver command** | Drop-down list for the selection of the command.
- **Start driver (online mode)** | Driver is reinitialized and started.
- **Stop driver (offline mode)** | Driver is stopped. No new data is accepted.  
  **Note:** If the driver is in offline mode, all variables that were created for this driver receive the status switched off (OFF; Bit 20).
- **Driver in simulation mode** | Driver is set into simulation mode.  
  The values of all variables of the driver are simulated by the driver. No values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
- **Driver in hardware mode** | Driver is set into hardware mode.  
  For the variables of the driver the values from the connected hardware (e.g. PLC, bus system, ...) are displayed.
- **Driver-specific command** | Enter driver-specific commands. Opens input field in order to enter a command.
- **Driver - activate set setpoint value** | Write set value to a driver is allowed.
- **Driver - deactivate set setpoint value** | Write set value to a driver is prohibited.
- **Establish connecton with modem** | Establish connection (for modem drivers) Opens the input fields for the hardware address and for the telephone number.
- **Disconnect from modem** | Terminate connection (for modem drivers)
**Show this dialog in the Runtime** | The dialog is shown in Runtime so that changes can be made.

### DRIVER COMMANDS IN THE NETWORK

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

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

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

10.1 Analysis tool

All zenon modules such as Editor, Runtime, drivers, etc. write messages to a joint log file. To display them correctly and clearly, use the Diagnosis Viewer (main.chm::/12464.htm) program that was also installed with zenon. You can find it under Start/All programs/zenon/Tools 7.50 -> Diagviewer.

zenon driver log all errors in the LOG files. The default folder for the LOG files is subfolder LOG in directory ProgramData, example:

%ProgramData%\COPA-DATA\LOG. LOG files are text files with a special structure.

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

In the Diagnosis Viewer you can also:

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

Note:

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

In case of an error, the Errorlogfile SAIA2nd32_xxx.txt is created. In this file the source of error is logged,

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 100 / 129...228</td>
<td>SAIA internal error codes.</td>
</tr>
<tr>
<td>110/238</td>
<td>No connection.</td>
</tr>
<tr>
<td>111/239</td>
<td>Connection not possible.</td>
</tr>
<tr>
<td>112/240</td>
<td>Number of objects for bits more than one object.</td>
</tr>
<tr>
<td>113/241</td>
<td>Number of objects for Uchar more than one object.</td>
</tr>
<tr>
<td>114/242</td>
<td>Type is not supported.</td>
</tr>
<tr>
<td>115/243</td>
<td>Writing on input not possible.</td>
</tr>
<tr>
<td>116/244</td>
<td>Writing on output not possible.</td>
</tr>
<tr>
<td>117/245</td>
<td>Error code out of range.</td>
</tr>
</tbody>
</table>

10.3 Check list

Cable disconnected?
Wrong network parameters in the CP or PC?
Data block does not exist
Wrong data block size (too short)?
Too slow: Scattered data areas - increase update time.

For error analysis please send a project backup and the „error text file“ (in the project path RT\FILES\zenon\custom\log) to the responsible Support department.