Contents

1. Welcome to COPA-DATA help ........................................................................................................... 4

2. Driver simulation ............................................................................................................................... 4
   2.1 Simulation static ............................................................................................................................. 5
   2.2 Simulation - counting ...................................................................................................................... 6
   2.3 Simulation - programmed .............................................................................................................. 6
      2.3.1 Editor ........................................................................................................................................ 7
      2.3.2 Driver configuration .................................................................................................................. 13
      2.3.3 zenon Logic Workbench .......................................................................................................... 18
      2.3.4 Runtime ................................................................................................................................... 20
      2.3.5 Please not for variables in simulation projects ........................................................................ 31
      2.3.6 Error messages ......................................................................................................................... 32
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. Driver simulation
If the underlying process is not available when configuring, this can be simulated and tested in advance. Three modes are available for this:

- simulation - static (on page 5): constant values simulated by the driver
- simulation - counting (on page 6): values simulated by the driver are counted up
• simulation - programmed (on page 6): Values are calculated via a simulation project with zenon Logic

⚠️ Attention

*If the driver is stopped in mode Simulation - counting, only the counting is stopped. The variable is not switched to faulty. In all other modes the driver is really stopped.*

**Hint:** *Simulation - programmed* is not supported by drivers for:

- Internal variables
- Mathematical variables
- Simulator variables
- System variables

💡 Info

zenon variables which represent the zenon Logic IO variables are available in the project in state *Driver simulation programmed*.

💡 License information

Part of the standard license of the Editor and Runtime.

### 2.1 Simulation static

For a static simulation, no communication to the control is established; the values are simulated by the driver. In this mode the value remains constant. Values can be changed by the Runtime or the user. At a restart of the Runtime with *Simulation - static* theses values will however not be saved and are lost.
2.2 Simulation - counting

For a counting simulation, no communication to the control is established; instead, the values are simulated by the driver. In this mode, the driver increments the values within a value range automatically, starting with 0. If the maximum value has been reached, the counting process starts at 0 again.

Info

*For negative starting values, the counting process only starts at 0.*

*INT uses the maximum value from USINT to count.*

Attention

*If the driver is stopped in mode Simulation - counting, only the counting is stopped. The variable is not switched to faulty. In all other modes the driver is really stopped.*

2.3 Simulation - programmed

For Simulation - programmed, no communication is established to the PLC; instead, 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. It enables the status and time stamp of variables to be modified on the driver. Write commands to variables by zenon are forwarded to the simulation, redundant servers and standby servers are synchronized. This is how you also simulate complex processes.
To start the simulation program, select the Simulation - programmed mode in driver configuration (on page 13).

Info

Windows CE

*Simulation - programmed* is not available for Windows CE.

2.3.1 Editor

Create project

To create a program for Simulation - programmed, you must:

- select a single process driver
- make sure that the project name is valid.

Attention

Simulation - programmed is not supported by drivers for:

- Internal variables
Driver simulation

- Mathematical variables
- Simulator variables
- System variables

To create a simulation project:

- click in the group **Driver simulation project** in the property **Edit on Click here** -
- A new zenon Logic project is created.
- **Name and port numbers for Event port and Standard port** are automatically issued; change these as you wish
- the zenon Logic Workbench is opened

The simulation project's workshop is automatically closed if:

- the zenon editor is closed
- a driver is deleted and its simulation project is currently being edited in Workbench
- the simulation project for the driver is deleted via the **Delete property**
- the simulation project is renamed

**PROJECT ARCHIVING**

A zenon Logic project contains many files and folders. To manage them, in particular to simplify distributed engineering, all files are archived in compressed form in the **Simul_<Treiber-ID>.zip** file in the `<Sql Projekt Pfad>\FILES\zenon\custom\drivers`. This file is in the driver files in the editor. When starting zenon Logic Workbench, the files are automatically decompressed and compressed again when it is ended.

**CREATING VARIABLES**

When creating variables in a simulation project, the succession is important:

1. create the variable
2. select the valid data type in the zenon Logic Workbench
3. after that activate property **embed symbol**
**Background:** Variables are created with the **PLC marker** driver object type by default. This object type does not support all data types for all drivers. If a variable is copied in the zenon Logic Workbench with active property **embed symbol**, you must deactivate this property in order to change the data type. With this the variable is then deleted in zenon.

### Info

**Error message "Can not create variable"**

A variable cannot be created in the simulation object if the driver does not have driver object types - exception

driver variable - which support this data type.

On activation of embed symbol, the error message "can not create variable" is displayed.

### STRUCTURE DATA TYPES

At creating variable with a structure data type:

- they are created with the driver object type **PLC marker** if possible
- all driver object types with the exception of driver variable are tried until one can be used for all elements
- if no fitting driver object type is available,  
  - no variable is created for non-structure variables  
  - driver object type **PLC marker** is used for structure variables

### EXAMPLE:

A structure data type contains elements of type UINT and STRING. A variable is created for S7TCP driver and **embed symbol** is activated.

- The variable is not created in **PLC marker** but as Ext. Data module, in which all structure elements are present.

- A new type LINT is added; it is not supported by the Ext. Datablock.
When a new variable is created, it is created as type PLC marker. Only the first complex variable can be activated (UINT). The object type Ext. remains for the existing structure variable Data and the last structure element (LINT) cannot be activated.

COMPILING THE FILES

If a simulation project is compiled in zenon Logic Workbench, this causes a variable to be created in zenon and the Runtime program file. When compiling, a subfolder is created in Runtime-Folder\RT\FILES\zenon\custom\drivers with the name of the simulation project. The file with the code of the simulation is SIMULRT.COD is archived in this folder.

The following message is displayed in the output window:

SIMULRT.COD

And if compiled with the C compiler, then also:

TSAPP.DLL

⚠️ Attention

For multi-user projects (on page 11) you must not create simulation projects offline. They cannot be deleted anymore.

Delete project

To delete a simulation project:

1. click on the Click Here-> button in the Delete property in the Driver simulation project group
2. confirm this when requested to do so
3. the ZIP file with the project files is deleted
4. the zenon Logic Workbench is ended

Note:
the Runtime folder for the simulation project remains

function undo is not available for this action,

click on Edit to create a new simulation project

for multi-user projects (on page 11):

- the driver must be configured to Make changes possible so that the simulation program can be deleted
- the project is not displayed as deleted on other Clients as long as it has not been synchronized;

if you try to open a deleted project before it has been synchronized (click on Edit), a new simulation project will be created

- you cannot reverse the deletion of single-user projects via Cancel changes

⚠️ **Attention**

The Simulation project is immediately removed from the local database and the server database when deleted. This action cannot be undone!

**Note:** Manual deletion of the ZIP file of the driver also leads to the simulation project being deleted.

Requirement: The zenon Logic Workbench for this driver is not opened. We do not recommend doing it this way!

**Distributed engineering**

For multi-user projects, all running zenon Logic Workbenches that are part of the project are closed when:

- Accept changes for modules that are completely locked for other users in Enable changes, such as variables, drivers and data types

- Discard changes:
  - new projects are not created on the server
  - Changes in the simulation program are lost
  - Driver files not present in the server database are also lost in the local database

- Synchronize:
  changes made locally to the simulation program program are lost
Update local version:
changes made locally to the simulation program in the simulation program are lost

⚠️ Attention

The status of the simulation project’s ZIP file may not be additionally modified (Accept Changes, Enable Changes, Discard Changes multi-user status), to ensure correct Accept Changes and Enables Changes for the driver!

DELETE PROJECT

For multi-user projects

- the driver must be configured to Make changes possible so that the simulation program can be deleted
- the project is not displayed as deleted on other Clients as long as it has not been synchronized;
  - if you try to open a deleted project before it has been synchronized (click on Edit), a new simulation project will be created
- you cannot reverse the deletion of single-user projects via Cancel changes

⚠️ Attention

A simulation project with the status of Enable Changes cannot be deleted (on page 10) in Offline mode.

A simulation project created offline in a multi-user project can therefore no longer be deleted.

Change driver

For a driver change in zenon the following is true:

- The simulation project is maintained.
- All ports are however set to 0. To receive working port numbers, open the project in the Editor. At this new port numbers are entered automatically. Port numbers can also be assigned manually.
Attention

Drivers that are linked to a zenon Logic project using the integrated solution cannot be exchanged.

XML export/import

A project in mode Simulation - programmed can be exported via XML. It can however not be imported as new project for another driver.

Workaround:

1. Export the zenon Logic programs.
2. Import them to the new project for the new driver.
3. Replace the variables via search and replace.

2.3.2 Driver configuration

There are four modes available when configuring the driver:

- Hardware
- Simulation - counting
- Simulation static
- Simulation - programmed
Select the simulation - programmed mode.

Driver variable status

The status of zenon Logic is shown in the driver variables SimulRTState at Offset 60. The variable is numerical and cannot be written to. The value informs you of the status of Runtime:
<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>without application</td>
<td>Runtime has not loaded a program, the program was stopped or not loaded to Runtime.</td>
</tr>
<tr>
<td>30</td>
<td>not instanced</td>
<td>Runtime was not instanced, for example because the DLL with Runtime could not be loaded, there is too little memory, the DLL is not present or is the wrong version.</td>
</tr>
<tr>
<td>18</td>
<td>p-code available</td>
<td>p-code is available together with compiled code, switching is possible.</td>
</tr>
<tr>
<td>17</td>
<td>compiled code active</td>
<td>Runtime runs with compiled code (otherwise: interpreted p-code)</td>
</tr>
<tr>
<td>16</td>
<td>compiled code available</td>
<td>Compiled code from the C compiler is available</td>
</tr>
<tr>
<td>15</td>
<td>application is loaded</td>
<td>An application is present, Runtime is running</td>
</tr>
<tr>
<td>14</td>
<td>can’t start - missing handlers</td>
<td>Some &quot;C&quot; functions are missing</td>
</tr>
<tr>
<td>13</td>
<td>active breakpoints installed</td>
<td>At least one breakpoint was set by the debugger.</td>
</tr>
<tr>
<td>12</td>
<td>CT segment exists</td>
<td>Application was compiled with the &quot;complex variables in separate segment&quot;.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>(internal, for Runtime only.)</td>
</tr>
<tr>
<td>10</td>
<td>sysinfo request is available</td>
<td>(internal, for Runtime only.)</td>
</tr>
<tr>
<td>9</td>
<td>freeze event production</td>
<td>Binding does not transfer any events.</td>
</tr>
<tr>
<td>8</td>
<td>single cycle mode</td>
<td>(internal, for Runtime and debugger only.)</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>(internal, for Runtime only.)</td>
</tr>
<tr>
<td>6</td>
<td>locked variables</td>
<td>At least one variable is blocked.</td>
</tr>
<tr>
<td>5</td>
<td>trigo functions are in degrees</td>
<td>Trigonometric functions are stated in degrees.</td>
</tr>
<tr>
<td>4</td>
<td>log message(s) in stack</td>
<td>(internal, for Runtime only.)</td>
</tr>
<tr>
<td>3</td>
<td>application stopped between 2 progs</td>
<td>Runtime stopped between two steps during debugging</td>
</tr>
<tr>
<td>2</td>
<td>application stopped on SFC breakpt</td>
<td>(internal, for Runtime only.)</td>
</tr>
<tr>
<td>1</td>
<td>application stopped on error</td>
<td>Runtime stopped with a serious error, restart required.</td>
</tr>
<tr>
<td>0</td>
<td>application is running</td>
<td>TRUE = application is in &quot;run&quot; mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE = application pauses in &quot;cycle to cycle&quot; mode.</td>
</tr>
</tbody>
</table>
**Info**

The driver variable `SimulRTState with Offset 60` displays the status of Runtime on the server. This also applies if the variable is displayed on the standby server.

---

**Cyclical synchronization**

Simulations run in a network on the server and standby server independently of one another. When the standby server starts, there is a synchronization with the server, however:

- This first synchronization is postponed by the packet runtime
- Small differences in the execution lead to growing diversion

The status of the simulated variables also therefore becomes different. In order to keep these differences low, use the `Transfer driver simulation image to standby` to synchronize the simulation image:

- create a new function
- Select Transfer driver simulation image to standby

- The dialog to select the driver opens

- In the list, all process-connected drivers (with the exception of the simulator) are available (clicking on the column heading changes the sorting)

- Select the driver that is to be synchronized
Each time a function is called up on the standby server, an image is requested from the server for the selected driver.

The differences in the current simulation status can be minimized through cyclical activation. Adapt the grid of the execution to the extent of the data to be synced, because this places a load on the network and computer.

The execution of the function is set at Standby and fixed; it cannot be changed. It only happens if:

- The computer is the standby server
- The server is online
- The project is a network project
- Drivers were selected for syncing

**TO CHANGE THE SELECTION OF THE DRIVERS**

- In the driver properties, click in the General group in the Parameter properties
- The dialog for selecting the driver opens

**2.3.3  zenon Logic Workbench**

The Simulation—programmed is programmed in the zenon Logic Workbench, which works in close conjunction with the zenon Editor. For the simulation, (in contrast to the integrated solution) only the variables of a driver are exchanged between the editors.

**DEFINITION OF THE VARIABLES TO BE EXCHANGED:**

When starting zenon Logic Workbench, all driver variables that are not yet present in zenon Logic are created as global variables. The Embed symbol is automatically activated for these.

**VISIBILITY**

If a new variable is defined in Workbench in the global or retain area, this is only visible in zenon Logic. To make it visible in zenon too, the Embed symbols option must be activated. From this point in time, all variables are updated by both editors.
If the Embed symbol option is deactivated, the variable is deleted in the zenon editor. This action enables global or retain variables to be defined in zenon Logic, which are not visible for zenon.

For a newly-created variable created by the zenon Logic Workbench in zenon, an attempt is made to assign this to the SPSMERKER area. In the event that the zenon driver (on page 7) does not support the configured data type, a search is made for the first data area that supports the data type. If no area is found, the variable cannot be created.

A variable is only locked if in the case of a multi-user project and the enable changes status has not been set.

**PROJECT SETTINGS**

Some settings under Project -> Project parameters... -> Further options -> Extended in zenon Logic influence the functionality of the simulation.
## Extended settings

<table>
<thead>
<tr>
<th>Extended settings</th>
<th>Description</th>
</tr>
</thead>
</table>
| Runtime: execute mode | **Recommendation:**  
Mode: **triggered**  
cycle time: 1/4 to 1/3 of the driver update time  
for frequent value changes to simulated variables lead to increased load placed on the computer. |
| compiler/options: Embed symbols of all variables | **Recommendation:** Deactivate.  
**Active:** Data for a variable are exchanged, because the symbol names are assigned in Runtime.  
But: Changes to the configuration of the variables are not updated between zenon and zenon Logic. It is no longer possible to make changes online. |
| compiler/options: Retain capitalization of symbols | without effect: Variable names are always converted to capitals in Runtime. |
| compiler/options: Create status bits for variables with a profile | **Active:** The status of variables is simulated or reacted to in the simulation program. |
| "C" compiler | Native code is supported.  
Name of the DLL that the compiled code must contain: `t5app.dll`. |

### 2.3.4 Runtime

#### Functionality

zenon Logic Runtime offers the following functions:

- File transfer
- Data types:  
  - `BOOL`
- BYTE
- DINT
- DWORD
- INT
- LINT
- LREAL
- REAL
- SINT
- STRING
- TIME
- UDINT
- UINT
- USINT
- WORD

- zenon arrays (see also Arrays and simulation (on page 22))
- Date stamp/time stamp
- Digital recording of values
- Dynamically linked function blocks, stored in t5block*.dll
- Hot Restart
- Logging:
  - Trace messages from the program (printf)
  - Error messages in Runtime
  - all sent to Workbench and to the diagnosis server
- Machine code: Native compiled code with C post compiler (not available under Windows CE)
- Online Change
- Programming languages: IL, AS, FDB, LD, ST
- Retain Data (remanant Data)
- Special function blocks and features:
  - Data serialization
Arrays and simulation

To simulate multidimensional arrays at driver simulation, you must activate property 'Save complex variables in own segment' in the zenon Logic Workbench.

ARRAYS WITH START INDEX 1

If at a simulation arrays with start index 1 exist, these arrays are created with one more index in the zenon Logic Workbench.

Example

In zenon an array has indices 1 to 4. This array has the indices 0 to 4 in the zenon Logic Workbench.

This guarantees that the same indices are used in zenon and in zenon Logic. Index 0 is not transferred to the driver.
If you change array in the zenon Logic Workbench, you must consider this additional index.

**Runtime files**

When creating the Runtime files, a subfolder with the name of the simulation project is created in the Runtime folder of the project in the path `\RT\FILES\zenon\custom\drivers`. The file with the code of the simulation is `simulrt.cod` and is archived in this folder.

Once successfully compiled in the Workbench and transferred to the Runtime folder, the following message is shown in the output window of the Editor:

```
SIMULRT.COD
```

And if compiled with the C compiler, then also:

```
TSAPP.DLL
```

**REMOTE TRANSPORT**

The Runtime files for the simulation are created as a subfolder of the driver files. For remote transport, the and all information contained in it is transferred to the target computer.

**Data exchange**

The exchange of data from zenon Logic Runtime to the driver starts with the 2nd cycle of zenon Logic Runtime. Therefore the first run is available to initialize all variables in Runtime.

Data is only exchanged if:

- The variable is requested and:
  - did not yet receive a value
  - does not have the status `Switched off (OFF/_VSB_N_UPD)` or `Alternate value (ALT_VAL/_VSB_AVALUE)` in the driver
  - is still required by the driver.
  - has changed value or status

The exchange of data to the driver is carried out asynchronously to the Runtime cycle. The driver runs through the value changes once per update cycle. All values changed - with the exception of the `Switched off (OFF/_VSB_N_UPD)` or `Alternate value (ALT_VAL/_VSB_AVALUE)` status - are entered into the list of variables to be updated in the driver and sent to Runtime.
TRANSFER DRIVER SIMULATION IMAGE TO STANDBY FUNCTION

This function is only executed on the standby server (for configuration, see Cyclical comparison (on page 16) chapter). The data is synchronously fetched from the driver on the server that is compiling the image. The driver has 5 seconds of time for this. Because the data for the image is created in the same thread in which the simulation is running, it must be ensured that the simulation is processed within 5 seconds. If this time is exceeded, no image is transferred for this driver.

VALUE SIMULATION

Value changes are not transferred immediately, but stored in a buffer. The size of the buffer amounts to 8192 value changes or the five time the number of variables, according to whatever value is greater. The delay in transferring resulting from the driver cycle can be up to 100 ms. If the value changes again whilst it is waiting to be transferred, all values that have not yet been transferred are sent to Runtime and the new value is then noted. This can lead to an increased load for the computer and network, but ensures that all values are transferred in Runtime.

DRIVER WRITE COMMAND

If simulation - programmed is configured and active, the write commands at standby are passed through in Runtime.

Write operations are carried out by the driver asynchronously to Runtime. Data is exchanged via a buffer. The size of the buffer amounts to 8192 value changes or the five time the number of variables, according to whatever value is greater.

Write commands are ran through after their values are changed in Runtime. If a write command cannot be executed correctly, the error message Write queue full! Write command for <DP-Name> lost! is displayed.

If the status bits are activated, setting Writing successful will display if the value was written from the driver to the value of the variable. The Runtime program should delete this status again after processing the write command. A status bit that is already active does not cause any delay in the driver writing again. Write commands are also executed if the Runtime program was stopped by a breakpoint.

Note: Not all write commands from the driver must be visible in Runtime. If several write commands are in the buffer, only the last one is visible in Runtime.

SWITCH OFF VARIABLES, SWITCH TO SUBSTITUTE VALUE

If the variable is switched off or switched to a substitute value, this is not recognizable for Runtime. Changes in values that are also carried out by Runtime are not assigned to the variables in the driver. If
the variables are again switched to spontaneous value or activated, the current value is again carried over from Runtime to the driver. When turning off the spontaneous value, status bit OFF is set. When you switch back to the spontaneous value, the OFF bit is reset.

Data storage

RETAINE DATA

Retain data is stored in the SimulRt<TreiberID>.ret file. One file such as this is created per simulation, provided retain variables are present. The file is taken into account when a comparison is carried out in the network.

DRIVER REMANENT

The Simulation - programmed setting at the driver start is not a criterion in order to restore the remanant variable image.

Redundancy

START ON STANDBY SERVER

If the driver is already running on a standby basis, then the image for the simulation should already be available. In this case, the simulation is started "hot" with this image. If there is no image, a warm a start is carried out.

TRANSFER DRIVER SIMULATION IMAGE TO STANDBY FUNCTION

This function is only executed on the standby server (for configuration, see Cyclical comparison (on page 16) chapter). The data is synchronously fetched from the driver on the server that is compiling the image. The driver has 5 seconds of time for this. Because the data for the image is created in the same thread in which the simulation is running, it must be ensured that the simulation is processed within 5 seconds. If this time is exceeded, no image is transferred for this driver.

COMPARISON

In addition to the program files of the simulation, the following are also compared:

- Retain data (*.ret)
Driver simulation

- all *.simul files
  This file extension is created by the simulation program for simulation specific files and archived in the folder with the computer description below the Runtime folder.

Status

STATUS SIMULATION

For the status simulation, in zenon Logic under Project -> Project parameters -> Extended -> Compiler -> Options -> Create status bits for variables with a profile must be active. If the status of a variable is modified, this triggers the transmission of a variable to the driver.

Changes to the following states does not trigger a value change at the driver and is also not taken over from the simulation:

- Real time external (T_EXTERN/VSB_RT_E)
- Real time internal (T_INTERN/VSB_RT_I)
- Standard time (T_STD/VSB_WINTER)
- Writing acknowledged (WR_ACK/VSB_WR_ACK)
- Writing successful (WR_SUC/VSB_WR_SUC)
- Normal status (NORM/VSB_NORM)
- Deviation from normal status (N_NORM/VSB_ABNORM)
- Select in the network (NET_SEL/VSB_SELEC)
- Runtime exceeded (TIMEOUT/VSB_RTE)
- In process (PROGRESS/VSB_DIREC)
- Switched off (OFF/VSB_N_UPD)
- Substitute value (ALT_VAL/VSB_AVALUE)

Info

State bit Writing successful (WR_SUC/VSB_WR_SUC) is available in the Runtime program and can be set back in order to display successful writing.

However only setting back the state does not trigger the transfer to the
STATUS BITS HANDLED DIFFERENTLY

The following status bits cancel each other out:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>17: Spontaneous (SPONT/_VSB_SPONT)</td>
<td>Status SPONTAN is set. This is the status set if status handling is turned off or no status was defined. Status GI + INAVLID is deleted.</td>
</tr>
<tr>
<td>18: Invalid (INVALID/_VSB_I_BIT)</td>
<td>Status INVALID is set. Status SPONTAN + GI is deleted.</td>
</tr>
<tr>
<td>16: General interrogation (GI/_VSB_GR)</td>
<td>Status General Interrogation is set. Status SPONTAN + INAVLID is deleted.</td>
</tr>
</tbody>
</table>
| 8: Select in the network (NET_SEL/_VSB_SELEC) | If this status bit is active (BSO activation/deactivation) the following status bits are accepted with 0 and also transmitted as 0 to the driver to identify a change:  
- Select in the network (NET_SEL)  
- Cause of transmission (COTx)  
- Select (SE_870)  
- N_CONF (P/N-BIT)  
- Test bit (TEST)  
- Writing successful (WR_SUC) |

SELECT BEFORE OPERATE (SBO)

Status simulation must be available in the Runtime program in order for Select Before Operate to be reacted to. The procedure for Select Before Operate is defined in Runtime.

ACTIVATION

If an SBO select is sent to the driver, it triggers a value being written with state Select in the network (NET_SEL) + cause of transmission (COT_act).
A corresponding SBO procedure must be implemented and must start in Runtime.

The state bit Select in the network (NET_SEL) must be set back in the Runtime program.

**DEACTIVATION**

The deactivation triggers the writing of a values with status Select in the network (NET_SEL) + Cause of transmission (COT_deact).

The SBO procedure must be ended in Runtime accordingly.

The state bit Select in the network (NET_SEL) must be set back in the Runtime program.

**Start / stop**

**DRIVER START**

The value updating only starts at the standby server if the driver has received the process image from its server. The image must be received within the time out module, otherwise the driver will start without an image. If the standby server is upgraded to the server within the waiting waiting time, waiting is also ended. Drivers that start in a network project or without a network have no waiting time. This behavior applies for both the simulation as well as for the hardware mode.

**RUNTIME START**

If the driver starts the simulation - programmed, the first stage is loading the DLL with the simulation Runtime. After this, Runtime is parametered and the Runtime program starts, provided it can be successfully loaded.

If an image of Runtime is present, Runtime is started hot with this. Otherwise, a warm start is carried out. If this is also not possible, an attempt is made to start Runtime cold. If there is no valid program, Runtime boots up stopped. The status of all variables is set to INVALID.

The simulation program runs in it own thread and is therefore completely independent from the driver cycle.
Info

Retain variables

*Retain data contain only the value of the zenon Logic variables not their status. This means for the start:*

- Warm start: The status which was set for a variable is restored - regardless of whether it is a retain variable or not.
- Cold start with retain variables: Only the value of the retain variable in zenon Logic is restored, not the status.

STOP RUNTIME

If Runtime is stopped, the status INVALID is set for all variables. Variables that are requested when Runtime is stopped have the status INVALID as initialization. Runtime secures the data for the warm start and online change:

- **SIMULRT.HOT**: contains the data for the hot restart.
- **SIMULRT.UPD**: contains the data for the online change.

Both these files are created in the Runtime program folder. Runtime must have write authorization for this folder.

RELOAD

- Recompiling the simulation project: causes the corresponding driver to be reloaded.
- Modification of the Time Out module: is not recognized as a change by Runtime and does not cause the driver to be reloaded.

Standard implementation in the driver kit triggers a hot restart when reloading the simulation project. If this is not possible (for example because the program is different), a cold start is carried out with retain.

REMOVING RUNTIME

If Runtime is running, this is stopped. Before Runtime is removed, all outstanding value updates are sent to the driver. Only then is Runtime released and SimulRuntime.dll removed from the PC.
Driver commands

Driver commands are used to influence drivers via zenon, e.g. start and stop. The engineering is implemented with the help of function Driver commands.

STOP/START DRIVER

If the driver is stopped in simulation - programmed mode, this will result in Runtime being removed from the memory. All variables obtain status INVALID. When the driver is restarted, the simulation is reloaded and Runtime is started.

SWITCH HARDWARE/SIMULATION DRIVER

When switching between modes Hardware and Simulation, the driver behaves according to the following pattern:

1. If the driver was configured for hardware mode in the Editor:
   - the driver is set to mode Simulation static after function Driver command is executed with parameter Driver in simulation mode.

2. If the driver was configured in the Editor for one of the simulation modes (static, counting, programmed:
   - the driver is set to hardware mode and communicated with the control after function Driver command is executed with parameter Driver in hardware mode.
   - the driver then changes back to the configured simulation mode after function Driver command was executed with parameter Driver in simulation mode

Linking variables

The zenon driver is allocated to the zenon Logic variables via the names converted into capital letters. If the zenon Runtime requests a variable from the driver, it forwards this to the simulation. If no simulation is loaded, the variable receives status Switched off (OFF/VSB_N_UPD) when the driver is stopped. Otherwise it receives status Invalid (INVALID/_VSB_I_BIT). Variables of driver variable object type are never requested by the simulation.
time stamp

To use the time stamp, in zenon Logic under Project -> Project parameters -> Extended -> Compiler -> Options -> Statusbit create for variables with profile must be activated.

If in the Runtime program, the date for a variable is set, the value of the date and time is used as the time stamp. When the date or time changes during the last cycle, this always triggers a transfer of the value to the driver. This also applies if neither the value nor the status have changed.

Most drivers only transfer a new time stamp if the value and/or status change at the same time. An example of an exception is the IEC870 driver, which also transfers new timestamps from the hardware (or Simulation - programmed) with the same value and same status to Runtime.

Variables stamped by the Runtime program receive the status Real time external (T_EXTERN/_VSB_RT_E).

All variables to be transferred to the driver that are not stamped by Runtime receive a joint time stamp in the cycle. This ensures that the time corresponds to that of the change. These drivers receive the status Real time internal (T_INTERN/_VSB_RT_I).

DATE & TIME

The following applies for all date and time information:

- Times are UTC.
- All times must be between 02. 01. 1970 and 2038.
- Dates are converted into a string in the format YYYY/MM/DD.
- The time is converted into the format HH:MM:SS.

2.3.5 Please not for variables in simulation projects

The following is true for the creation and modification of variables in simulation projects:

1. zenon Logic
If variables are created for a simulation project in the zenon Logic Workbench, they do not have a valid addressing for the communication. If the communication is switch to hardware, the communication can be interrupted when the new variables are used in screens without adjusting the addressing correctly.

2. **Rename**

If a variable is renamed in a simulation project, the name of the variable is also changed in zenon accordingly. So for example you must adjust the names of variables which are used in a VBA project also in the VBA code.

3. **Integrated solutions**

Variables of integrated solutions such as zenon Logic, IEC870 or IEC850 must not be renamed in the simulation project. In this case the variables lose their project information. For example: If you rename `Project0/Global/NewVar` to `myNewVariableNewVar` in the simulation project, it becomes `myNewVar` in zenon. After the renaming a communication with the zenon Logic Runtime is impossible.

4. **Communication based on variable names**

Variables of a driver which communicates based on the variable names must not be renamed in the simulation project.

5. **Deleting or importing variables**

If a variable is deleted or imported while online change is activated in zenon Logic, all variables are removed and inserted again in zenon Logic.

6. **Variables with active "embed" symbol**

Variables which exist in a program and whose "embed" symbol is active are not available via the Shared Memory interface in zenon. If a variable with an active "embed" symbol is moved from the global area to a program, the variable is deleted in zenon.

2.3.6 **Error messages**

Error messages in zenon editor:
<table>
<thead>
<tr>
<th>Error message</th>
<th>cause and solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The chosen name was already used for another driver or contains invalid</td>
<td>The new name of the simulation project is already used in the project or is not</td>
</tr>
<tr>
<td>characters! Please select another name.</td>
<td>valid.</td>
</tr>
<tr>
<td></td>
<td>▶ Give it a valid name</td>
</tr>
<tr>
<td>The simulation project from the <code>&lt;File name&gt;</code>/<code>&lt;Description&gt;</code> driver has not</td>
<td>A simulation project is available for the driver in Workbench. This was not</td>
</tr>
<tr>
<td>been compiled and cannot be supported! Please compile the project in</td>
<td>compiled however.</td>
</tr>
<tr>
<td>Workbench.</td>
<td>▶ Start the zenon Logic Workbench with this driver's simulation project</td>
</tr>
<tr>
<td></td>
<td>▶ compile the project</td>
</tr>
<tr>
<td>Write queue full! Write command for <code>&lt;Name Variable&gt;</code> lost!</td>
<td>Displays the loss of a write command.</td>
</tr>
<tr>
<td></td>
<td>Tip: It does not make a difference if the diver settings are set to <code>simulation</code> -</td>
</tr>
<tr>
<td></td>
<td><code>programmed</code> in the driver settings</td>
</tr>
</tbody>
</table>