HIMA

SMART SAFETY.

Programming Tool

SILworX®

Release Notes V12.28

Contact

HIMA Paul Hildebrandt GmbH

P.O. Box 1261

68777 Brühl, Germany

Phone: +49 6202 709-0

Fax: +49 6202 709-107

E-mail: info@hima.com

Document designation	Description
HI 801 544 D, Rev. 1.02 (2025)	German original document
HI 801 550 E, Rev. 1.02.00 (2026)	English translation of the German original document

Table of Contents

1 SILworX V12.28

This chapter describes the improvements, new functions and restrictions of SILworX V12.28 compared to the previous versions.

1.1 Compatibilities

This chapter describes the compatibility of the individual software versions with the hardware versions and Windows operating systems.

1.1.1 PADT Operating System

As of V11, SILworX is a 64-bit application and is only supported by Windows 10 (64-bit).

1.1.2 Supported HIMA System Families

SILworX V12.28 can be used for the following HIMA system families:

- HIMax.
- HIMatrix F systems.
- HIQuad X.

In general, the TÜV version lists of the individual system families must be taken into account.

1.1.3 Not Supported HIMA System Families

As of SILworX V9.36, the following controllers are no longer supported:

- HIMatrix F10 PCI 03.
- HIMatrix F20 01.
- HIMatrix F30 01.
- HIMatrix F31 02.
- HIMatrix F31 03.
- HIMatrix F35 01.
- HIMatrix F60 01.

SILworX versions as of V9.36 no longer support these controllers (exclusion of liability). This applies to code generation, download and online services.

This means that projects in which the specified controllers are used, may still be opened in SILworX, e.g., for update or upgrade purposes (replacement by successor products). However, connections to the specified controllers are no longer permitted with SILworX versions as of V9.36.

1.1.4 Compatibility with Previous Projects

SILworX V12.28 can convert and edit projects that were created with a previous version.

When further developing SILworX, HIMA is committed to ensuring that code generation involving a new SILworX version will only result in modified configuration CRCs in exceptional cases. These cases are documented in the release notes of each SILworX Version.

1.1.5 Compatibility with PADT Hardware

HIMA recommends using modern computer hardware. In particular with very large projects, old PCs may require long processing times and thus be inappropriate for this task. Therefore, state-of-the-art computers should be used whenever possible. Enhanced hardware features such as computing power (number of CPU cores) and memory space (RAM) result in improved performance.

1.1.6 Windows Permissions for Installing and Operating SILworX

When installing and operating SILworX under Windows, observe the following:

- Administrator permissions are required to perform the installation.
- User permissions are sufficient to start or operate SILworX.

2 New Functions

- OPC UA is a new functionality introduced in this version.

- The values for *Minimum Configuration Version* were extended by *SILworX V12*, e.g., to support the OPC UA Server functionality. New default setting: SILworX V12.

- SILworX API offers a wide range of functions for operating SILworX. Libraries (e.g., for Python and C#) enable users to integrate API commands into their automation environment.

  Some of the available functions are:

  - Creating projects.
  - Opening new projects, also including the user management.
  - Closing projects.
  - Quitting SILworX.
  - Reading SILworX status, e.g., the version or licenses.
  - Creating child nodes from the Project structure tree.
  - Retrieving the structure tree.
  - Performing validations in the structure tree.
  - Archiving objects such as the project or structure tree nodes.
  - Restoring objects such as the project or structure tree nodes.
  - Generating code, e.g., for download or reload.
  - Connecting or disconnecting resources.
  - Logging in to modules.
  - Starting and stopping resources.
  - Starting and stopping systems and their modules.
  - Loading configuration into resources through downloads.
  - Reading online values (at system and module level, individually or in groups).
  - Querying diagnostic entries of modules.
  - Performing downloads of operating systems.
  - Starting and stopping global forcing.
  - Reading and writing to global force variables.

  General Information about API:

  - The use of SILworX API requires a license.
    SILworX API must be explicitly activated in the *settings.ini* file.
  - Access to the SILworX API is only possible via SSL (TLS 1.2).
    This requires the installation of OpenSSL and a valid certificate.
  - Access to projects via the SILworX API requires the same user permissions as during human interaction.
  - Configurable timeouts when accessing the SILworX API ensure that projects are automatically closed if no further API queries are sent within the timeout.
  - Any API activity is displayed in the SILworX status bar.
  - Any actions are tracked in the SILworX logbook. This applies to both human interaction and API accesses.

  Important:
  Users must perform a tool classification and qualification for their SILworX API application.

  The API documentation in HTLM format and a C# application example is available in the subfolder …\c3\openapi within the SILworX installation directory.

- The following resource types are no longer supported:
  - HIMatrix F10 PCI 03
  - HIMatrix F20 01
  - HIMatrix F30 01
  - HIMatrix F31 02
  - HIMatrix F31 03
  - HIMatrix F35 01
  - HIMatrix F60 01

  Projects including these resource types can still be opened. No new hardware can be created for these resource types. Code can no longer be generated for these resource types. Online connections to controllers with these resource types can no longer be established. If projects that include these resource types are opened, a warning is displayed for each resource impacted.

- The functions *Undo* and *Redo* are now available in the FBD Editor.

- If the update of all instances or value fields aborts with an error, all elements being updated are reset to the state before the action. In previous SILworX versions, elements that could be updated without errors, were not reset despite the aborted update.

3 Improvements

- Generating code for X-OTS connected to an X-OPC Server Set with two servers fails if system variables of the X-OTS connection to the second server are used.
  This problem was removed.

- SILworX issues a reload warning, if a reload code generation is performed after the COM configuration of a HIMatrix controller has changed.

- OLT field contents did not fit within the field size. Texts that are longer than the OLT fields are shortened, which is indicated to the user with an ellipsis … at the end of the field. To display the full text, users can drag the OLT field larger. Alternatively, the text can also be viewed via tooltip.

- SILworX also allows GSDML libraries to be stored as a child nodes of libraries. So far, GSDML files could only be stored as a child node in exactly one fixed GSDML library subordinate to a PROFINET controller. It is now possible to share GSDML files across controllers or resources, which avoids redundancies.

- Offline simulation: A user program or a cycle step can be started again if the following conditions apply:

  - The SQRT function with REAL data typing is only used in the user program within user-defined function POUs.

  - When executing the SQRT function, the IN input has a negative REAL value, e.g., -0.5.

  This means that NaN is issued to OUT and the ENO output is set to FALSE in all uses of the SQRT function.

- English and German texts that include units are corrected to comply with DIN 5008:
  1mA -> 1 mA
  1°C -> 1 °C
  1mV -> 1 mV
  24V -> 24 V

  This change may cause the columns of newly exported CSV files to differ from those in existing files. It might only be possible to re-import existing CSV files after replacing the texts, e.g., using the *Search* and *Replace* functions.

- The *OC Blanking (Number of SC/OC Intervals)* parameter in the X-DO 24 01 and X-DO 24 02 modules specifies the number of test intervals (parameter *SC/OC Interval [µs]*). For further details refer to the HIMax release notes (HI 801 549 E).

4 Restrictions

The use of SILworX is subject to restrictions. If the following instructions are observed, the restrictions have no influence on the safety and availability of the code generated for a controller.

4.1 FBD Editor

- Effect:
  In the FBD Editor, comment texts in assigned comment fields are no longer displayed after new pages have been added.

  Condition:
  The following conditions must apply:
  - A page with a comment in the assigned comment field is located next to a second page that contains at least one logic element and
  - on the second page, the action *Insert Empty Pages -> Insert Column* or *Insert Empty Pages -> Insert Row* is executed such that the new pages are inserted between the page with the comment field and the second page.
  The text in the comment field is temporarily no longer displayed.

  Workaround:
  The text reappears if the assigned comment field is moved or the editor is closed and then reopened.

- Effect:
  In the FBD Editor, page information is not correctly positioned after importing a project from ELOP II.

  Condition:
  The following conditions must occur simultaneously:
  An associated comment or OLT field is located on an empty page without further logic elements, while the element connected to it is located on a different page.

  Workaround:
  Associated comment or OLT fields should be located on the same page as the elements connected to it.

- Effect:
  Conflict icon for variables remains visible, in spite of fixed conflict.

  Condition:
  In the following cases, the conflict icon remains visible although the invalid action was canceled and the valid value displayed:
  - Invalid name is entered in the local variable field.
  - An interface variable is assigned an existing sequence number.

  Workaround:
  Start the verification or update process.

- Effect:
  Information on global variables used as VAR_EXTERNAL is not displayed:
  If global variables with Struct or Array data types are used as VAR_EXTERNAL, the FBD Editor does not display the information entered in the columns *Initial Value*, *Description*, *Additional Comment*, and *Technical Unit* for the sub-elements.

  Condition:

Create a global variable with *Struct* data type and enter a description in a *Struct* element. Use this global variable as a value field in a program. Then view the created VAR_EXTERNAL in the *Local Variables* tab.

Workaround:
View the attribute properties of the VAR_EXTERNAL in the corresponding global variable.

- Effect:
  Empty pages in the drawing area of the FBD Editor cannot always be deleted.

  Condition:
  The *Delete Empty Pages* context menu option is not active if the following conditions occur simultaneously:
  - A line extends over two or more adjacent pages of the empty page.
  - The line does not cross the empty page.

  Therefore, the empty page cannot be deleted.

  Workaround:
  None.

- Effect:
  In rare cases, using the *Undo* or *Redo* function may cause a POU instance in an FBD function block to become invalid.

  Condition:
  - In the FBD Editor, use drag&drop to copy a function block from the Object Panel into the drawing area. This creates a POU instance that references the function block.
  - Change a POU instance property (e.g., set the Retain attribute) and save the contents of the FBD Editor. The asterisk (*) in the header of the FBD Editor disappears.

  Note: The error only occurs after the contents of the FBD Editor have been saved. If the contents are not saved and the asterisk is still displayed in the FBD Editor header, the POU instance reference remains valid.

  - In the Object Panel, rename the function block used (i.e., referenced). The new function block name applies immediately.
  Perform *Undo*. In doing so, the POU instance reference is reset to the previous function block name. However, the actual renaming action is not undone causing the POU instance to become invalid.

  Workaround:
  Manually reset the name of the referenced function block to the previous name. In doing so, the reference is restored.

- Effect:
  To undo instance names and their comments 2 *Undo/Redo* steps are necessary.

  Condition:
  In the Instances tab, change the instance names and comments so that they contain some identical text sections. Use *Search* and *Replace* to replace the identical text sections with another text. To undo the changes in the instance names and comments, perform *Undo* twice.

  Workaround:
  Perform *Undo* twice.

4.2 Structured Text

- Effect:
  2700 consecutive comment lines are not possible in the ST Editor.

  Condition:
  SILworX terminates when commenting out 2700 consecutive lines in the ST Editor.

  Workaround:
  Partition long comments, e.g., by grouping 1000 lines to one comment.

4.3 Hardware

- Effect:
  The Detail View button in the Hardware Editor is not active.

  Condition:
  Start the Hardware Editor online view and open a module's detail view: The *Detail View* button is not active.

  Workaround:
  Not required, as the detail view contains a *Close* button.

- Effect:
  During code generation, SILworX as of V6 no longer stores the licenses sorted by entry order, but by name.

  Condition:
  Enter the license names in non-alphabetical order in V5 and generate the code. Then generate the code in V6.

  Workaround:
  Use suitable license names, ask for HIMA technical support.

- Effect:
  The global error statistics are only displayed in the higher-level controller and not in the remote I/Os. The system variables for *Forcing*, *CPU Autostart Enable*, *CPU Start Enable*, *CPU Main Enable*, *ReadOnlyInRun* and *Start Cycle* are not activated by the remote I/Os. The system variables of a remote I/O are not automatically evaluated by the higher-level controller.

  Condition:
  Assign the system variables to global variables and simulate the corresponding errors.

  Workaround:
  Use the error statistics of the higher-level controller. Assign global variables to the system variables of the remote I/O error statistics and evaluate them in the logic.

4.4 User Program

- Effect:
  The following behavior of the EXPT function in PES does not comply with the IEEE-754 standard.
  1.0 ** NaN := 1.0 expected: NaN
  EXPT.ENO := TRUE expected: FALSE

NaN ** 0.0 := 1.0 expected: NaN
EXPT.ENO := TRUE expected: FALSE

EXPT in OTS and in the offline simulation behaves in compliance with IEEE-754.

Condition:
See above.

Workaround:
If ENO is required, trap or avoid a NaN on both inputs.

- Effect:
  The DIV_TIME function from the standard library improperly sets the ENO error output to FALSE and therefore reports an error under the following conditions:
  - The IN2 input (divisor) is of type REAL.
  - The value of IN2 is +/-INF.

  Condition:
  Use DIV_TIME with EN/ENO and enter +/- INF as the divisor. (INF is the result of 1.0 / 0.0, for example).

  Workaround:
  Ignore ENO in this case.

- Effect:
  During the offline simulation and OTS, the EXPT function provides the result NaN instead of 1.0, if IN1 = 1.0 is used for the basis and IN2 =-INF is entered for the exponent.

  Condition:
  Call up EXPT with IN1 = 1.0 and IN2 = -INF (or another large negative number) and view the result in the offline simulation or OTS.

  Workaround:
  If this special case is relevant for the application, this has to be processed in the user program.

- Effect:
  A POU is processed in accordance with the following sequence: first the sequences, afterwards the SFC actions, and then the FBD logic. As a result, the input values of SFC transitions and SFC actions that are described in the FBD logic always originate from the previous cycle. The specific evaluation of the input values, however, reveals small differences:
  During the FBD processing, the input value of an SFC transition is written to and retained in the SFC transition memory and only processed in the sequence during the next cycle. After a cold start, this has the effect that sequences generally do not move on to the next step before the second cycle.
  The input value of an SFC action is read from the source during the processing of the SFC action. If this is a function, the initial value is read since functions are initialized at the beginning of POU processing and are only processed after the SFC actions.

  Condition:
  See above.

  Workaround:
  SFC transition:
  When programming sequences, users must take into account that an SFC transition is performed in the second cycle at the earliest.

SFC action:
To use a function result as input value for an SFC action, a variable must be connected between function output and SFC action input.

- Effect:
  Sequences of a branch ending with SFC steps result in a deadlock. Sequences of a simultaneous branch ending with transitions, result in several active steps outside the simultaneous branch.

  Condition:
  See above.

  Workaround:
  Users must take suitable measures to ensure that such faulty sequences are not used.

- Effect:
  Access to an array element with an index outside the range of values result in accessing an element of the array based on a defined and high-performance procedure, to avoid random access to memory areas.

  Condition:
  See above.

  Workaround:
  Using suitable programming, users must ensure that the indexes used to access the array elements are within the value range of the array.

- Effect:
  Various elements of a structure variable cannot be written from different sources.
  The user program and the hardware or communication cannot write to different elements of the same structure variable.

  Condition:
  See above.

  Workaround:
  Use different structure variables for the elements written to by the user program and for the elements written to by the hardware or communication.

- Effect:
  Elements of variables of a user-defined data type cannot be used as array indexes.

  Condition:
  See above.

  Workaround:
  Copy the value of the required variable to a standard variable and use this as the index.

- Effect:
  Conversion of floating point numbers has changed.
  When floating point numbers are converted from the decimal format to the corresponding binary format for the data type (REAL, LREAL), SILworX V11 obtains a different result than SILworX V10. The reason for this behavior is a more modern conversion algorithm, which is supposed to operate faster and more accurately.

  The new results are used in every configuration generated with SILworX as of V11 (program,

ke.config, etc.) and may also become effective with it.

The version comparison between a configuration generated up to V10 and a configuration generated as of V11 indicates a CRC change for each file and POU affected by it.
For an affected value field, the POU detail view reports a change in the associated instance or assignment.
For an affected local variable, the POU detail view reports a change in the initial value CRC of the POU.
For an affected global variable, no message is issued concerning the change initial value (in the detail view of ke.config).

For a force value, the *Edit Force Data* dialog box immediately displays the result of the conversion from decimal to the binary and back to decimal format, so that users can view it before sending.

So far, only one concrete case has been observed: SILworX V10 converts the literal 2.2250738585072012e-308 into the LREAL value with the bit pattern 16#000F_FFFF_FFFF_FFFF, which corresponds in SILworX to the decimal format 2.2250738585072009e-308.
SILworX V11 converts the literal 2.2250738585072012e-308 into the LREAL value with the bit pattern 16#0010_0000_0000_0000, which corresponds in SILworX to the decimal format 2.2250738585072014e-308.
The two results only differ by one unit in the last place (ulp). The exact literal value deviates approx. 0.37 ulp from the new result and approx. 0.63 ulp from the old result. For the decimal conversion within certain value ranges, which also include this literal, the IEC 559:1989 standard allows the result to slightly deviate from the optimal approximation, so that both results appear permissible (0.63 ulp < (0.37 ulp + 0.47 ulp)).

Condition:
Example based of the case considered: Enter the literal 2.2250738585072012e-308 into a value field or enter it as the initial value of a local or a global variable with LREAL data type. Connect the output of the value field to another value field or a POU instance appropriately. Generate code for this example in both SILworX V10 and V11. In SILworX V11, perform a version comparison between the V10 configuration and the V11 configuration.
Either load the respective configuration into the PES or start the offline simulation in SILworX V10/V11 and observe the effective values in the Force Editor.

As online alternative, enter the same literal as the force value of a variable and observe the displayed result.

Workaround:
If each bit is important, enter a REAL or LREAL literal in such a way that it comes as close as possible to the real number assigned to the required bit pattern[1]. For an unambiguous conversion, up to 9 significant decimal places are required for REAL and up to 17 for LREAL (total number before + after the decimal point, without consideration of the exponent).

[1] The bit pattern of a binary floating point number is mathematically assigned to a very specific real number. At the same time, the floating point number is a proxy for all real numbers in an interval around the number assigned to it (in many cases, but not always, symmetrically).

4.5 Version Comparison

- Effect:
  During the version comparison, the detail view of a POU incorrectly shows the change to a POU instance if the two following points apply:
  1. The comparison base was created with a version prior to V4.116.
  2. The name of the called-up POU type contains umlauts.

Condition:
See above.

Workaround:
Only use spaces and characters from the following list for function block names:
- 0 1 2 3 4 5 6 7 8 9
- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- a b c d e f g h i j k l m n o p q r s t u v w x y z
- $ % & ( ) * + - / : ; < = > ? \ ^ _ ` { | }

- Effect:
  Once completed the version comparison of a configuration generated with a SILworX version prior to V9 and a configuration generated with SILworX as of V9, the message "The order of the variable and instance declaration has changed" no longer appears for the following elements:
  1. Function blocks and functions.
  2. Programs that use at least one of the following standard functions in the configuration generated with SILworX as of V9: ADD, SUB, MUL, DIV, MOD, MOVE, AND, OR, XOR, NOT, SHL, SHR, all ANY-to … functions, ADD_TIME, SUB_TIME, MAX, MIN, SEL, MUX, GT, LT, GE, LE, EQ, NE, PACK.

  Condition:
  Create a resource with two configurations (first configuration: loaded or imported, second configuration: generated) in a SILworX version prior to V9. A comparison of these versions (correctly) notifies a change to the declaration order in a POU.
  The project state has to match the generated configuration.
  Convert the project to SILworX V9 and generate the code.
  The version comparison will not show the declaration order message, even though the respective POU displays the same CRC difference as before.

  Workaround:
  If the project state matching the comparison base (e.g., the loaded configuration) is still available:
  1. Convert a copy of this old project state to SILworX V9 or the required new version. Generate the code there and export the result via the start dialog box of the version comparison.
  2. Open the other project to be used for the version comparison in the new SILworX version, import the previously exported configuration into the start dialog box of the version comparison and use it as a comparison base.

4.6 Code Generation

- Effect:
  Changing the CONST attribute of global variables after they have been used in the user program results in a conflict.
  If a global variable is used as VAR_EXTERNAL and then the CONST attribute is set or reset, code generation will result in a conflict when the VAR_EXTERNAL is assigned a value, but the CONST attribute of the global variable is TRUE.

  Condition:
  Use a global variable as VAR_EXTERNAL in the logic and change the CONST attribute of the global variable.

  Workaround:

Delete the global variable in all locations in which it is used. This removes the associated VAR_EXTERNAL. Then reinsert the global variable in all locations.

- Effect:
  In very rare cases, the program CRCs resulting from code generation performed with SILworX as of V5.30.0 and with a previous SILworX version are different.

  The version comparison between two configurations - a configuration generated with the SILworX version as of V5.30.0 and one with a previous version - detects a CRC difference in the respective program binary file (*.ldb or *.dll) and associated files (program parameters, ls.config, root.config). However, the subordinate data type, POU type and global variable CRCs are the same. No messages appears in the detail views of the program file and the POUs to explain the difference.

  The cause and effects of the difference cannot be determined from these symptoms.

  Condition:
  Create a project with a SILworX version prior to V5.30.0 and generate code for the resources
  Convert a project copy to a SILworX version as of V5.30.0 and generate code again.
  Compare the configurations generated in the 2 SILworX versions.

  Workaround:
  None.

- Effect:
  In a user program's memory layout, the global variables that the user program uses for reading and writing are arranged in 4 adjacent groups:
  (a) Global variables with no Retain attribute.
  (b) Multi-source global variables[1] with no Retain attribute.
  (c) Multi-source global variables[1] with Retain attribute.
  (d) Global variables with Retain attribute.

  At the intersections between these groups there may be padding (unused bytes). The reload code generation incorrectly issues a reload warning if there is no padding between (a) and (b) or between (c) and (d) during reload.
  The reload warning designates the deleted (padding) element by *x.10* or *x.20*" where x is the name of the user program.

  [1] In addition to the user program, multi-source global variables have at least one other source in a protocol (in SILworX V9: Modbus slave V2).

  Condition:
  Example: Configure a resource with a protocol of type *Modbus Slave Set V2*.
  Create a global variable A of BOOL data type and a global variable B of DINT data type.

  Edit the predefined Modbus slave data view and drag global variable B (DINT) onto its *Register Inputs*. Save the action.

  Edit the program, drag global variable A and global variable B onto the logic. Then, draw a line from the output to the input for each of the value fields created in this way (feedback loops). Save the action.

  Generate code and load the result (or export and re-import it in the version comparison).

  Remove the value field for global variable B, including its lines, from the program logic. Save the action.

  Generate code for performing a reload with the loaded (or imported) configuration as

baseline.

Workaround:
In this case, the warning can be ignored.

4.7 Reload

- Effect:
  1. If *Abort* is selected when assigning the comparison configuration, an incomprehensible error message may be displayed.
  2. When generating reload code for several resources and importing a config file, the user prompts only resume the reload code generation once the config file has been confirmed.

  Condition:
  Start the reload code generation for a configuration with multiple resources that include a backup or import configuration and abort in the selection dialog box.

  Workaround:
  None.

- Effect:
  Naming a newly inserted POU instance the same as a POU instance that was previously deleted affects the reload process, sometimes critically:

  Case 1: After reload is completed, the new POU instance starts with the last process values (and force data) of the POU instance that had the same name and was deleted. This may produce unexpected results and induce an unintended behavior of the system.

  Case 2: Transferring the process values to the new function instance has no consequences since function instances are always assigned their initial values before they are executed. (Reading their outputs before their execution always provides the initial values as well). Also, function instances have no force data.

  Case 3: The reload code generation reports an error.

  Condition:
  Case 1:
  - A is a user-defined function block type.
  - B is a standard or user-defined function block type.
  An instance of B is deleted from the logic of A and later, a new instance of B is inserted in the same or another place of the logic of A.
  The problem occurs if the user unintentionally names the new instance the same as the deleted one without considering the consequences resulting from this action. The instance name is first automatically assigned by the PADT and can be subsequently changed by the user, in which case the name of the deleted instance can be reused.
  No differences exist between the new and deleted instances, which would be rejected by the reload process.
  - For this logic change, the user performs the reload code generation and the reload.

  Case 2:
  - A is a user-defined POU type or a program.
  - B is a function type.
  - Otherwise as case 1.

  Case 3:
  - As case 1 or case 2, but the differences between the new and the deleted instances are not allowed by the reload process.

Workaround:
After code generation, case 1 and case 2 can be recognized as follows: In the detail view of the POU type within the version comparison, the new instance is not marked as *New*, but as *Changed* (and/or as *Moved* with respect to execution sequence), or it does not appear at all.

4.8 Protocols

- Effect:
  Incorrect default value for the timing master after project conversion to V6

  Effect:
  In a project prior to V6, create a safe**ethernet** connection with 2 resources (Res1, ID = 1 and Res2, ID = 2) and set the connection to *V6 and higher* in a current version. This may lead to Res2 becoming the timing master.
  From the user's perspective, it is not possible forecast which of the 2 resources will be the timing master when converting to *V6 and higher*.

  Workaround:
  Set the timing master explicitly.

- Effect:
  After conversion to SILworX V10 or a higher version, the safe**ethernet** connections in the project are disrupted.
  Only a question mark *?* appears instead of the system ID.
  Example: *?.0.1 (192.168.1.206:6010)*.

  Condition:
  The problem occurs when converting projects prior to SILworX V10 if connections across configurations contain / in the resource name of the X-OPC Server.
  Open the *Properties* dialog box for the safe**ethernet** connection and check the *IF Channel 1* and *IF Channel 2* fields in the *Partner* tab.

  Workaround:
  safe**ethernet** connections can be corrected by setting the partner in the overview of the safe**ethernet** connections first to *None* and then to *New*.

4.9 Project

- Effect:
  Projects can get lost on the network drive due to the Windows internal synchronization of network drives. The project may then only be available locally.

  Condition:
  Open, edit, and close a project in a synchronized folder. Afterwards, the project is removed from the network drive.

  Workaround:
  Do not open projects that are being synchronized and where there is a network connection. Opening a project leads to its network version being deleted by Windows. In this case, copy the project saved in the local drive back to the network drive.

- Effect:
  If a combination of a non-breaking hyphen and a space is used in the program name, SILworX can terminate.

Condition:
In the structure tree, create a program *P 1* as a child node of the resource. Then create another program *P 1* child node of the same resource, but this time use Alt+0173 to set a non-breaking hyphen in the name before the space. SILworX then terminates.

Workaround:
None.

- Effect:
  In SILworX V4, delete actions could cause objects that could no longer be edited to remain in the database. These objects did not affect the project, but were reported during the project integrity check.

  Condition:
  Projects created in SILworX V4 and V5 that contain such residual objects most likely cannot be converted to SILworX V6 and V7. The likelihood is particularly high if the projects contain user-defined data types.

  Workaround:
  Remove the objects found during the integrity check prior to converting the project. To this end, perform the following steps in the previous SILworX version:
  1. Archive all child nodes of the project that are positioned in the structure tree under the project, except for *Programming and Debugging Tool*.
  2. Create a new project in the previous SILworX version.
  3. Delete the *Configuration* node in the new project.
  4. In the new project, restore the configuration archived in step 1 and, if existing, additional child nodes of the project.

  The project just created should be convertible to the current SILworX version.

- Effect:
  After conversion to SILworX V10 or a higher version, mono safe**ethernet** connections are invalid.
  Error message due to non-resolvable references to objects that were deleted in a version prior to V10. These objects reappear after the conversion and must be removed manually. This could be a deleted processor module or communication module, for example, which was used in a safe**ethernet** *IF Channel*. In V10, this channel will show a *?* reference to the module that is no longer available. Otherwise, a path to the referenced object is displayed.

  Condition:
  In a SILworX version prior to V10 (e.g., V9.36.0), create a new project with two resources. Add a processor module and a communication module to each resource. Create a safe**ethernet** connection between the resources and use both modules. Then delete the communication module or the processor module.
  The reference to the channel seems to have disappeared. After this step, convert the project to V10. The reference is still displayed with a *?* as system ID because the target cannot be resolved.

  Workaround:
  Manually correct the reference or set it to *None*. For a second safe**ethernet** *IF Channel*, deactivate the second channel manually.

4.10 Documentation

- Effect:
  Cross-references of *Struct* or *Array* elements do not appear in print.

Condition:
Use a *Struct* or *Array* element, check the cross-references in the documentation and compare them with the cross-references in SILworX.

Workaround:
None.

4.11 User Management

- Effect:
  When restoring a user management archive created with V9, a default user contained in the archive is ignored. Users must explicitly log in with user name and password.

  Condition:
  - Create a new project with user management.
  - Define the default user.
  - Archive the user management.
  - Remove the existing user management in the project.
  - Restore the archived user management.

  Workaround:
  None.

4.12 Licenses

- Effect:
  U3ID dongles may become invalid when upgrading to a newer Windows version or updating within a Windows version (mainly long-running Windows versions such as Win10).

  Condition:
  Operate SILworX with a U3ID dongle. When updating a Windows version or upgrading to a newer Windows version, e.g., from Win7 to Win10, the dongle may become invalid because the U3ID dongle is suddenly read with a different value.

  Workaround:
  A new U3ID dongle must be created, in particular for newer Windows versions.

- Effect:
  The message in the logbook when requesting a license is misleading.

  Condition:
  Request the license.

  Workaround:
  Upload the file *C:/Users/Administrator/Downloads/silworx_lic_request_3467898754.olciml* to the HIMA Extranet. For assistance if problems occur, contact: support@hima.com.

4.13 SILworX API

- Effect:
  The *PADT User Management* project tree node cannot be archived using the SILworX API, even if the user is logged in as *Security Administrator*.

  Condition:

If users are logged in to the SILworX API with *Security Administrator* access and have projects open, they cannot archive the *PADT User Management* project tree node.

Workaround:
None.

- Effect:
  The binary stability of archives is not ensured in the following cases:
  - After converting projects between different SILworX versions.
  - If reference targets are changed so that the referencing positions must be adjusted.
  - When changes are saved and then undone within an editor.

  Condition:
  - Convert projects between different SILworX versions.
  - Change reference targets, e.g., rename a data type. This implicitly changes all global variables that use this data type.
  - Save the changes and undo them.

  Workaround:
  None.

5 Special Points

When using SILworX, the described characteristics must be observed.

- Effect:
  In the Hardware Editor, the scaling settings for an analog value are read as REAL. SILworX reads the values specified for the vertices of an analog value as REAL (at 4 mA and 20 mA). They are, however, further processed as LREAL. LREAL can also be used in the user program. This restriction is only relevant with very large or very small vertex values.

  Condition:
  Using extremely small or extremely large vertex values can impair process value accuracy.

  Workaround:
  Process raw values in the user program.

- Effect:
  Logic operations of BOOL variables having values that originate from third-party systems can provide results that differ from those expected.

  Condition:
  The cause is that the coding of BOOL values used in the third-party system deviates from the coding used in the HIMA system.

  Workaround:
  Two workarounds are possible:
  - The external system only provides 0 for FALSE and 1 for TRUE.
  - A correction circuit is implemented in the user program for all relevant BOOL variables to normalize the value to 0 or 1:
  Non-normalized variable -> AtoByte function block -> AtoBOOL function block -> normalized variable.

- Effect:
  The cycle times can strongly vary during calculations with variables of data types REAL or LREAL, particularly when using trigonometric functions.

  Condition:
  See above.

  Workaround:
  To measure the watchdog time, the cycle time must be determined under realistic conditions. For further information on how to determine the watchdog time through testing, refer to the corresponding safety manual.

- Effect:
  The value of user program's system variables is not displayed during the online test and offline simulation:
  - The OLT field is empty.
  - The value of digital system variables is not represented by the color of the corresponding line.
  The *Process Value* column in the *System Variables* tab of the Object Panel is empty.
  - The Force Editor contains no system variables.

  Condition:
  See above.

  Workaround:

Most of the information is displayed elsewhere, e.g., in the Control Panel. To display it in the OLT field, connect the system variable to a variable and connect this variable to an OLT field.
Forcing is only possible if the system variable is connected with a variable.

- Effect:
  SILworX handles value changes of VAR_INPUT variables in user-defined function blocks differently
  In user-defined function blocks, VAR_INPUT variables differently, depending on how the inputs are wired:
  - If the inputs are connected to variables of a standard data types, the value of the variables is transferred to a copy within the function block (call by value).
  - If the inputs are connected to variables of a user-defined data type, a reference to the variable is transferred to the function block (call by reference).

  Condition:
  This behavior may result in errors if all the following conditions are met:
  - The source of a VAR_INPUT variable is a VAR_EXTERNAL variable.
  - The same source of the VAR_INPUT variable is simultaneously used in the called function block as VAR_EXTERNAL variable.
  If the value of the VAR_EXTERNAL variable is changed in the function block, the subsequent reading of the corresponding VAR_INPUT variable in the function block results in the following actions:
  - For a user-defined data type, the current values are read.
  - For an elementary data type, the previous values, which were valid at the beginning of the function block instance processing, are read.

  Workaround:
  VAR_EXTERNAL variables should not be used simultaneously as the source of a VAR_INPUT variable for instances of this POU.

- Effect:
  The document management cannot print the contents of the online help associated with a user-defined POU.

  Condition:
  None.

  Workaround:
  Use Windows to display the online help contents and print out the individual topics.

- Effect:
  It cannot be ensured that key terms in the export or import files (.CSV, .XML) do not change between SILworX versions. If this occurs, SILworX imports the corresponding data as default values and issues an error message.

  Condition:
  Example: The data type for the English language setting was denoted *Data Type* in versions prior to V5, and *Data type* in V5 and higher. When an export file is imported from a version prior to V5, SILworX creates all the variables with the default data type BOOL.

  Workaround:
  Adjust the corresponding key words in the file to be imported.

- Effect:
  If the diagnostic view is opened during a system login and the connection is closed, SILworX offers the module login when attempting to re-establish the connection.

Condition:
Log in to the hardware.
Open the detail view for the module.
Open the diagnostics for this module in a second window.
Then close the connection.

After a lost connection, the diagnostics only offers the module login.
The detail view only offer the system login.

Workaround:
Once the module login dialog box for the diagnostics has been opened, the diagnostics and module online view must be closed and opened again. Only then can they be read again via the system.

6 Upgrading from a Previous Version

Project data from previous SILworX versions can still be used in V12.28.

When further developing SILworX, HIMA is committed to ensuring that code generation involving a new SILworX version will only result in modified configuration CRCs in exceptional cases. These cases are documented in the release notes of each SILworX Version.

HIMA supports upgrades as of V6.114. HIMA no longer supports versions prior to V6.114.

Observe the following procedure to upgrade from a version as of V6.114 to V12.28:

1. Create a backup (archive) of the existing project.
2. Generate code for all resources.
3. Export all codes and import them in the version comparison. Deviations resulting from the conversion can be detected in the version comparison
4. Create a new backup (archive).
5. Open the project in V12.28 and convert it.
6. After completing the conversion, check the project integrity.
7. Generate the code in V12.28 to detect potential errors. Perform a version comparison with the imported configuration and check whether CRCs have changed.
8. Remove detected errors and re-generate the code to detect CRC changes.
9. If no CRC changes are detected, the upgrading was completed successfully.
10. If CRC changes are detected, verify whether they can be accepted. If the changes can be accepted, the upgrading is successfully completed.
11. If the changes cannot be accepted, continue to work with corresponding previous version.

Conversion notes:

- The conversion can take some time for very large projects.

6.1 References

- SILworX online help.
- SILworX first steps manual, HI 801 103 E.
- Communication manual, HI 801 101 E.

Release Notes

HI 801 550 E

For further information, please contact:

HIMA Paul Hildebrandt GmbH
Albert-Bassermann-Str. 28
68782 Brühl, Germany

Phone: +49 6202 709-0
Fax +49 6202 709-107
E-mail: info@hima.com

Learn more about SILworX online:

www.hima.com/en/products-services

www.hima.com