

Programming Tool

SILworX®

Release Notes V10.58

All of the HIMA products mentioned in this manual are trademark protected. This also applies for other manufacturers and their products which are mentioned unless stated otherwise.

HIQuad®, HIMax®, HIMatrix®, HIQuad X®, SILworX®, XMR®, HICore® and FlexSILon® are registered trademarks of HIMA Paul Hildebrandt GmbH.

All of the technical specifications and information in this manual were prepared with great care and effective control measures were employed for their compilation. For questions, please contact HIMA directly. HIMA appreciates any suggestion on which information should be included in the manual.

Equipment subject to change without notice. HIMA also reserves the right to modify the written material without prior notice.

All the current manuals can be obtained upon request by sending an e-mail to:
documentation@hima.com.

© Copyright 2018, HIMA Paul Hildebrandt GmbH

All rights reserved.

Contact

HIMA Paul Hildebrandt GmbH

P.O. Box 1261

68777 Brühl

Phone: +49 6202 709-0

Fax: +49 6202 709-107

E-mail: info@hima.com

Document designation	Description
HI 801 498 D, Rev. 10.00 (1820)	German original document
HI 801 499 E, Rev. 10.00.00 (1828)	English translation of the German original document

Table of Contents

1	SILworX V10.58	5
1.1	Compatibilities	5
1.1.1	Compatibility with the PES Operating System	5
1.1.2	Compatibility with Existing Projects	5
1.1.3	Compatibility with the PC in Use	5
1.1.3.1	Use of Hardlocks	5
2	New Functions	6
2.1	HIQuad X	6
2.2	Smart Safety Test	6
2.3	Diagnostic Overview	6
3	Improvements	7
3.1	User Program	7
3.2	Cross-References	7
3.3	Structure Tree	7
3.4	Resource Properties	7
3.5	FBD Editor	7
3.6	Forcing	7
3.7	Optimized SILworX Database	8
3.8	Hardware	8
3.9	Protocols	9
3.10	System Configuration	9
4	Restrictions	10
4.1	FBD Editor	10
4.2	ST Editor	12
4.3	Cross-References	13
4.4	File Selection Dialog Boxes	14
4.5	Hardware	14
4.6	User Program	15
4.7	Version Comparator	17
4.8	Code Generation	18
4.9	Forcing	18
4.10	Reload	19
4.11	Smart Safety Test	19
4.12	Protocols	20
4.13	Project	21
4.14	Project History	22
4.15	Documentation	23
4.16	User Management	23
4.17	English SILworX	23
5	Special Points	24

6	Upgrading from a Previous Version	27
6.1	References	27

1 SILworX V10.58

This document describes the improvements and new functions of SILworX V10.58 compared to the previous versions.

1.1 Compatibilities

1.1.1 Compatibility with the PES Operating System

SILworX V10.58 can be used for the following HIMA system families:

- HIMax
- HIMatrix F systems
- HIMatrix M45 (discontinued)
- HIQuad X

1.1.2 Compatibility with Existing Projects

Version V10.58 can convert and edit projects that were created with a previous version. Generating code for an unchanged project does not cause the CRC to change.

1.1.3 Compatibility with the PC in Use

The minimum requirements for the computer used to run SILworX are specified on the corresponding HIMA DVD.

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 and memory space, result in improved performance.

1.1.3.1 Use of Hardlocks

The following points must be taken into account if SILworX is licensed on Windows 7, 8, 8.1, and 10 using hardlocks (USB sticks):

- Administrator rights are required to perform the installation.
- User privileges are sufficient for operation.

2 New Functions

2.1 HIQuad X

- Support for HIQuad X (H41X and H51X) in SILworX.

2.2 Smart Safety Test

- Users can create and automatically run test plans for their logic using the Smart Safety Test. It is possible to run the Smart Safety Test as a simulation in SILworX or on a controller. A test report in pdf format can be created after each test, as required.

2.3 Diagnostic Overview

- The current version contains a new *Diagnostic Overview* entry in the online hardware view for HIQuad X systems. It provides users with a table listing messages on the different system states during runtime. This information and the warning or error messages are generated according to predefined rules that process status information from the system or I/O modules to create user-friendly and practice-relevant messages.

3 Improvements

- This chapter describes the improvements of SILworX V10.x compared to versions prior to V10.x.

3.1 User Program

- For C++ POU, the proper stack size is used for X-CPU 31 after debugging.

If in a C++ POU, a different stack size is entered for X-CPU 01 and X-CPU 31, the user program's stack size will change, and thus its CRC. The version comparator indicates that the data size of the user program has changed. To generate the same user program as with the previous SILworX version, the user applies the X-CPU 01 stack size to the X-CPU 31.

3.2 Cross-References

- As of V10, the cross-reference view also includes cross-references to user-defined functions used in ST source texts (functions and function blocks).
- The cross-reference view for function blocks and functions has been improved. As of V10, it contains an individual entry for each use. Additionally, the *Info* column indicates the graphical position in the FBD logic, or the column/line number when used in ST. For each individual use, a Go to can be called up, which opens a target editor (FBD or ST) and selects the use.

3.3 Structure Tree

- Documentation can be called up using the context menu of objects selected in the project tree, but no longer via the *Project* menu. The *Documentation* options in the *Extras* menu and the action bar have remained unchanged and are still available.

3.4 Resource Properties

- The default setting for system bus latency is now called *System Defaults* instead of *Line Structure*.

3.5 FBD Editor

- In the FBD Editor, logic elements, such as variables, can be placed directly next to a comment field or an associated page comment field. This previously required a minimum spacing of 1 grid to a pin.
- The FBD Editor no longer shows any confusing yellow artifacts when zooming in on elements.

3.6 Forcing

- The force dialog box is more robust when variables are forced in quick succession. The system will no longer crash

3.7 Optimized SILworX Database

- SILworX performance has substantially improved in many areas thanks to a reorganization of the internal structure. Faster processing is especially noticeable when opening editors and copying large amounts of data. Opening and closing projects is now also faster.
- The *Reorganization* menu option has been eliminated completely. This function is now carried out automatically when the project is closed.
- References to other objects are now resolved as text. This results in a few changes: This can lead to further error messages.
 - The *Connect References* function has been eliminated. To connect an unconnected reference, it is sufficient to create an object with the corresponding destination name.
 - All references to these objects are adjusted when objects are renamed.
 - If the name of an object is changed, its cross-references will only be displayed again once the editor in which the name was changed has been saved.
 - If there are several destinations in the scope with the same destination name, one of them is selected as the destination. This selection is random for the user, but remains constant.
 - If a child variable of a global variable (with a user-defined data type) is used in function blocks, this use appears as cross-reference outside this function block, also for the parent variable.

Caution: If several objects have the same name, renaming an object causes all references to this object to change accordingly. The object that keeps the original name will then lose all its references.

Lessons learned:

HIMA recommends not renaming any objects unless the object name is unique.

- As the *Connect References* function has been eliminated, no more messages informing that references cannot be connected are generated during restoring, moving, or copying actions in the structure tree.
- Aborting the creation of documentation could lead to SILworX being terminated, particularly for large projects. The cause of this issue was identified and eliminated.
- In V9, project archives of previous versions containing user management could not be converted. Such project archives could only be opened as a project after they had been restored in the previous version. This intermediate step is no longer required.
- The internal data storage in SILworX has been restructured allowing objects within the editor to swap names.

3.8 Hardware

- Fieldbus interface names for COM modules correspond to the names printed on the enclosures: FB1, FB2, FB3.
- Fieldbus interface names for COM modules no longer include the modules' SRS.
- If the user configures a destination port without a source, a warning appears during validation and code generation that no mirroring will be carried out without definition of a source.
- For HIMax X-AI 16 51, the version comparator indicates changes to the process values *4 mA* and *20 mA* instead of the previous *Scal. Factor + Offset*.

- For ke.config, the version comparator now uses the module name + SRS instead of the previous *IO + SRS* designation to name HIMatrix I/O modules.
- When generating code for OPC Server Sets the condition *Target Cycle Time < Watchdog Time - 1000 ms* is now checked.

3.9 Protocols

- The data volume check for PROFIBUS DP has been reactivated. It had been temporarily deactivated in V9.
- References to service partners in the safe**e**thernet connection overview of the resource still apply after servers have been deleted. Verification messages notify users of any invalid references. Workaround:
No further actions are required after restoring the resource. Otherwise, the reference to the resource must be set to *None* in the safe**e**thernet settings.
- Whenever an IP link cannot be resolved, the system ID is replaced by a question mark, e.g., *?0.5 (192.160.0.1:6010)*.
- If an FB interface is used more often than permitted, the error message explicitly lists all protocols that use the FB interface. For Modbus slaves, the Modbus slave set now appears instead of the Modbus slave.

3.10 System Configuration

- Labeling of the code generation dialog for selecting the reload basis has been improved. The title now indicates the corresponding context to enable the resource to be assigned.
- The error message issued during a reload code generation when the data type has changed, offers a *Go to* function to the respective editor.
- When performing a reload code generation, users are informed of the number of required transfer operations and the maximum permitted transfer operations for a reload. This information helps users to better assess how close they are to reaching the maximum permitted number of transfer operations in order to improve planning of future actions.
- In previous versions, if a download or a reload was performed while the Force Editor or another online tool with force functions was open, the PADT issued the following error message:
The global force data query was aborted: <x>.
This PADT error message has been removed.

Remark: During download or reload, any force data displayed for a variable is not updated. This is indicated by grayed-out values in the Force Editor, etc., empty OLT fields and black power flow lines.

4 Restrictions

When using SILworX, observe the following 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:

Empty pages in the logic section of the FBD Editor cannot always be deleted.

Condition:

The *Delete Empty Page* context menu option is not active if the following conditions occur simultaneously:

- A line extends over two or more adjacent sides of the empty page.
- The line does not cross the empty page.

Therefore, the empty page cannot be deleted.

Workaround:

None.

- Effect:

In the FBD Editor, text in existing page comments is no longer displayed after inserting new pages.

Condition:

The following conditions must occur simultaneously:

- A page comment is located next to a second page with at least one logic element.
- 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 page comment and the page with the element.

The text of the existing page comment is temporarily no longer displayed.

Workaround:

The text reappears if the comment 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 main element is located on a different page.

Workaround:

Associated comment or OLT fields should be located on the same page as their main elements.

- 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 for a variable.
- - An existing sequence number is assigned to an interface variable

Workaround:

Start 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. Now 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:

Deleting a previously used POU and creating or inserting a new Typical using the name of the deleted POU leads to a system crash during code generation with this unsupported type.

Condition:

Create a POU *B* function block and use it in a program. Save and close all. Delete the POU *B* and create a Typical function block *B* there. Go where *B* is used and call up the update. The errors are rectified and the Typical is called up.

Workaround:

The change can be undone by using the correct type of function block. The correct type of instance can be checked by double-clicking the instance. The POU called up must not have the Typical type, which can be recognized by a small icon next to the name.

- Effect:

System crash when saving an FBD Editor if insertion of a logic part with connections is rejected during a copy action and the copy action is aborted. Users will not notice this immediately after a failed attempt to insert the copy, but only once they save the editor. Caution: Loss of data.

Condition:

Copy a logic part with connections (copy method is irrelevant), then insert it in a way that results in the copy action being rejected. The next time the editor is saved, this will most likely result in a crash.

Workaround:

Insert copied logic parts in a way that the insertion process is not rejected.

When working in the FBD Editor, save at regular intervals to minimize any potential data loss.

- Effect:

Deleting an OLT field, which is connected to a variable or block instance at a connection point, in the FBD Editor causes a wrong message to be issued during code generation if another connection exists at the connection point.

Condition:

1. Connect two variables in the FBD Editor.
2. Create an assigned OLT field at the outgoing connection point of the variable.
3. Delete the created OLT field.
4. Save the changes in the Editor.
5. Start the code generation.

Workaround:

Delete the connection line outgoing at the connection point of the deleted OLT field's variable, and then reconnect it.

Go to on the error message jumps to the affected object.

▪ Effect:

Renaming function blocks leads to conflicts in the logic's inputs and outputs that have the previous name of the function block. When resolving these conflicts, lines to those inputs and outputs are deleted.

Condition:

- Create and open a function block. Then, create an output variable named *OUT* in the editor.
- Create a second function block named *OUT*.
- Use the first function block in the program.
- In the next step, rename the second function block from *OUT* to *Block_2*.
- Reopen the program or start the verification process. A message informing that the reference of the instance output to 'Block_2' could not be resolved is issued.

Workaround:

- Rename the function block interface having the same name as the function block.
- Refresh the conflicts in the logic.

4.2 ST Editor

▪ 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.

▪ Effect:

Copying a user-defined data type from one variable to another within an ST function block editor leads to a system crash if the data type selection box of the target variable is in editing mode.

Condition:

Open an ST function or an ST function block and create 2 variables (any variable type). Assign a user-defined data type to one variable, then select this data type in the selection box and select *Copy* in the context menu. Then go to the data type for the second variable and press *Enter* to switch the selection box to editing mode. Insert the previously copied data type while in this mode.

Workaround:

Copy a selected, user-defined data type from one variable to another within an ST function block editor without switching the data type selection box of the target variable to editing mode, i.e., without pressing the Enter key or double-clicking it.

4.3 Cross-References

- Effect:

If variable cross-references are used in HIPRO-S, no structure information is displayed; the field remains empty.

Condition:

Use a variable in a HIPRO-S connection and view the cross-references in the Global Variable Editor: The *Structure Info* column remains empty.

Workaround:

None.

- Effect:

Display error in cross-references.

Contents of the columns *Info*, *Structure Info* and *Use* may be obsolete for a cross-reference if they contain information that is modified outside the associated editor. This affects, for example, the system ID for hardware cross-references or the name of a structure tree element.

The *Structure Path* column in the cross-reference view for global variables lists the correct name and path. The *Go to jump* is also not affected by the display issue.

Condition:

For example, using a global variable in a function block:

Create a new FBD function block.

Create a global variable and use it in the FBD function block.

Then rename the function block or copy and insert it.

In the Global Variable Editor, check the cross-references of the global variable. The *Info* column shows the old name of the function block while the *Structure Path* in the last column is correct.

For example, using a global variable in the hardware:

Assign a global variable to a system variable within a hardware rack and save it.

Then change the system ID of the resource and view the cross-reference of the global variable in the Global Variable Editor.

Workaround:

After renaming a structure tree element, open the editor that was used to create the reference, make an arbitrary modification (e.g., move a function block just a little) and save. This action updates the cross-reference information and the correct content is now displayed in the *Info* column. The same workaround can be used in the Hardware Editor.

- Effect:

Cross-references are shown in the language of the editor used for the reference and not the language currently set in SILworX.

Condition:

Create cross-references that contain language-dependent parts. Then open the project with a SILworX instance in another language.

The cross-references still contain the texts in the previous language.

Example: Names of system variables in the Hardware Editor are shown in the cross-reference of the global variables, but if they are linked in a German SILworX version, they will still remain in German in the English SILworX version.

Workaround:

- a) Ignore the issue.
- b) Save the editor where the link exists, e.g., by using *Go to* on the cross-reference (to be able to save the editor, make a (small) modification in this editor). This regenerates the cross-reference information (and updates them to the current language).

4.4 File Selection Dialog Boxes

- Effect:

After an invalid file selection, the hardware XML import dialog box remains open and only responds to the Cancel button.

Condition:

Select an incorrect or invalid file path in the hardware XML import dialog box and start the import.

Workaround:

Terminate the dialog box with Cancel and restart the import.

- Effect:

The restore dialog box shows only the path for the last restored archive files. The list of the recently created archives is missing.

Condition:

Open or create a project. Create one or more archives for the project. In the restore dialog box, open the selection box under archive files: This is empty; no other archive files are listed.

Workaround:

Select the archive file by browsing to the archive path.

4.5 Hardware

- Effect:

The detail view toggle button in the Hardware Editor is not active. The view can be closed using the Close button.

Condition:

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

Workaround:

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

- Effect:

During code generation, SILworX V6 and higher no longer stores the licenses sorted by entry order, but by name. This may result in a changed CRC when converting projects from previous versions.

Condition:

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

Workaround:

Use suitable names; ask for HIMA technical support.

4.6 User Program

- Effect:

The following behavior of the EXPT function in PES does not comply with the IEEE-754 standard.

$1.0 ** \text{NaN} := 1.0$, expected: NaN

EXPT.ENO := TRUE, expected: FALSE

$\text{NaN} ** 0.0 := 1.0$, 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 MUL function block provides faulty values if the following conditions occur simultaneously:

- HIMatrix standard resource (e.g., F30 01, F35 01, F60 01)
- Data type LREAL.

Input IN1 has the value $\pm\infty$, input IN2 has the value NaN (not a number).

- In this case, the result is $-\infty$, and not NaN as specified.
- In this case, ENO provides the correct result, i.e., FALSE.

Condition:

See Effect.

Workaround:

Ignore the result in this specific case.

- 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 $\pm\text{INF}$.

Condition:

Use DIV_TIME with EN/ENO and enter $\pm\text{INF}$ 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 $\text{IN1} = 1.0$ is used for the basis and $\text{IN2} = -\text{INF}$ 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 programmed accordingly in the user logic.

- 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 array elements are only accessed through indexes 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 two 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 index.

Condition:

See above.

Workaround:

Copy the value of the required variable to a simple variable and use this as index.

4.7 Version Comparator

- Effect:

The detail view of a POU in the version comparator incorrectly shows a change for a POU instance if:

1. the comparison base was created with a version prior to V4.116 and
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:

When comparing a configuration generated with a SILworX prior to V9 and a configuration generated with SILworX V9 or higher, the following message no longer appears: *The order of the variable and instance declaration has changed.*

1) For all function blocks and functions.

2) For programs that use at least one of the following standard functions (in the configuration generated with V9 or higher): ADD, SUB, MUL, DIV, MOD, MOVE, AND, OR, XOR, NOT, SHL, SHR, all ATO... function blocks, ADD_TIME, SUB_TIME, MAX, MIN, SEL, MUX, GT, LT, GE, LE, EQ, NE, PACK.

Condition:

Create a resource with two configurations (first config loaded or imported, second config generated) in a SILworX version prior to V9. A comparison of these versions (correctly) notifies a change in 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.

4.8 Code Generation

- **Effect:**

Conflict resulting from changing the constant attribute for global variables after their use: A conflict occurs during code generation, if a global variable is used as VAR_EXTERNAL and is set from Constant to Changeable or vice versa, when a value is assigned to this VAR_EXTERNAL and the global variable is constant.

Condition:

Use the global variable as VAR_EXTERNAL in the logic and change the constant state in the global variable.

Workaround:

Delete the global variable at all positions where it is used so that VAR_EXTERNAL disappears. Then insert it again at all positions.

4.9 Forcing

- **Effect:**

The force data edit dialog box displays an incorrect force status if more than one program exist and the action does not refer to the program shown in the first tab (from left) in the Force Editor. This means that the force data edit dialog box always (incorrectly) displays the force status of the program in the first tab from the left for all programs located in tabs 2 to 32 in the Force Editor. This is a particular issue if the force data is not edited in the Force Editor but in the online FBD Editor, as the force status is only displayed in this force data edit dialog box here, and no additional, correct force status is displayed in the background above the variable table.

Condition:

There are at least two program tabs in the Force Editor.

The force status displayed in the force data edit dialog box is only correct for the first tab from the left for local forcing, no matter whether the edit dialog box has been opened in the Force Editor or the online FBD Editor. When editing force data for any other program tab, the edit dialog box continues to display the force status from the first tab. The user thus receives incorrect and misleading information.

Workaround:

Edit local variables in the Force Editor or in the Watchpage, as the correct force status for the respective program is displayed in the force table in the background here, unlike in the online FBD Editor.

- Effect:

No new global or local variables with a name containing a special number suffix according to the following structure can be used on the Watchpage: `_0#` where `#` represents one or more numbers, e.g., *Program.Pou1.Var_01*, *Global Variable_02*. When such variables are inserted into the Watchpage table via drag&drop, the new elements are provided with a changed name. Names are changed as follows, for example: *Global Variable_02* --> *Global Variable_2*.

This fault has both safe and unsafe (misleading) effects.

Safe effect:

No force variable can be addressed with the incorrectly changed name.

Unsafe effect:

An unintended force variable can be addressed with the incorrectly changed name.

Condition:

Global or local variables with the following suffix (number suffix) in the identifier exist that are used within the resource configuration or user logic: `_0#`, where `#` represents one or more numbers and `0` can comprise several zeros.

Open a new Watchpage for global or local forcing online or offline.

Insert variables with number suffix into the Watchpage table via drag&drop. Watchpage variables created in this way have differing names and therefore address different or non-existent force variables (status: unconnected).

Workaround:

The user can correct the names of new Watchpage variables in the Watchpage table.

4.10 Reload

- Effect:

A reload code generation leads to a reload warning if the partner's download info cannot be detected:

*safe**ether**net reload sequence started. A dual configuration has been created. No download configuration is available for "[partner name]" to align the safe**ether**net signatures.*

This also occurs if the reload code generation was started without prior changes.

Condition:

Establish the **safeether**net connection to the proxy resource. Perform the reload code generation on the real controller without any changes.

Workaround:

None.

4.11 Smart Safety Test

- Effect:

If the mentioned conditions apply, calculating the lower limit of the tolerance range (setpoint - tolerance value) as ULINT results the limit to be underrun. The lower limit actually applying is then $((2 \text{ raised to the power of } 64) + (\text{setpoint} - \text{tolerance value}))$.

In the very probable case that the applying lower limit is greater than the tolerance range

upper limit (which the PADT properly calculates as ULINT in saturation arithmetic), the comparison always issues the status NOT_OK, irrespective of the actual value.

Condition:

In a CHECK_VALUE test plan step for a global variable with data type USINT, UINT, UDINT or ULINT, enter a setpoint with tolerance value, where tolerance value > setpoint such that (setpoint - tolerance value) < 0.

Example: Data type USINT, setpoint property "12 [14]": (12 - 14) = -2 is less than 0.

Workaround:

Modify the setpoint and tolerance value as follows:

New setpoint = (previous setpoint + previous tolerance value) / 2, rounded up or down to better match the test case.

New tolerance value = new set point.

The new lower limit is then 0, the new upper limit is within the previous upper limit range ± 1 .

4.12 Protocols

- Effect:

Create a safe**ethernet** connection with two resources (Res1, ID=1 and Res2, ID=2) in a project prior to V6 and set the connection to *V6 and higher* in a current version. This may lead to Res2 becoming the timing master.

Condition:

From the user's perspective, it is not possible forecast who will be the timing master when converting to *V6 and higher*.

Workaround:

Set the timing master explicitly.

- Effect:

Attempting to adopt an array variable with more than 32,768 elements into a communication protocol causes SiLworX to terminate.

Condition:

Create a global variable with 32,800 sub-elements and include it in a protocol as a process variable.

Workaround:

Partition large array variables into several smaller parts.

- Effect:

A fault is reported during verification if the maximum permissible data volume of a PROFIBUS DP slave is exceeded.

Condition:

The process data for a PROFIBUS DP slave must exceed the maximum permissible data volume:

Input data volume > 192 bytes (for HIQuad H41X or 51X > 244 bytes).

Output data volume > 240 bytes (for HIQuad H41X or 51X > 244 bytes).

Perform the verification or code generation to create the message.

Workaround:
Observe the limits.

4.13 Project

- Effect:

Projects can get lost on the network drive during Windows internal synchronization of network drives. They 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 the project leads to the network version being deleted by Windows. If this issue occurs, the local copy can be copied back onto 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:

Create a program P 1 under the resource in the structure tree. Then create a program *P 1* under the same resource, using Alt-0173 to create a non-breaking hyphen in the name before the space. SILworX then terminates.

Workaround:

None.

- Effect:

In SILworX V4, deletion actions could cause objects to remain in the database, but be no longer editable. These objects did not affect the rest of 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. The simplest procedure is described below and must be performed 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:

Error message due to destinations that cannot be resolved after project conversion to V10. References to objects that were deleted prior to V10 reappear after the conversion and must be removed manually. This could be a deleted CPU or COM module, for example, which was used in a **safeethernet** interface 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:

Use a version prior to V10 (e.g., V9.36.0) to create a new project with two resources and add a CPU and a COM to each. Create a **safeethernet** connection between the resources and use both modules. Then delete the COM or CPU module.

The reference to the channel seems to have disappeared. Now convert to V10. The reference reappears (with a ? as system ID, as the destination cannot be resolved).

Workaround:

Manually correct the reference or set it to *None*. For a **safeethernet** interface: Manually deactivate the second channel.

- Effect:

Archives created with SILworX versions prior to V10 cannot be restored in SILworX V10 if the archived object has a name that is not a valid file name in Windows, particularly if it includes invalid characters (`\:*?" <>)<>`).

Condition:

Create a structure tree object in a SILworX version prior to V10 with an invalid file name (a:b or Test/01). Archive the object. An archive is created. The proposed name for this archive replaces the invalid characters with an underscore.

Any attempt to restore this archive in V10 fails.

Workaround:

Open the archive in the original version (or in V9), rename the object, and archive it again.
OR

Restore the archive in the original version and archive the parent object. The parent object can then be restored in V10 and the required object can be copied or moved.

4.14 Project History

- Effect:

When an English project created with SILworX V2 is imported, SILworX does not properly interpret the date in the project history. Example: 1/11/2013 is interpreted as November 1, 2013 instead of January 11, 2013. 1/13/2013 is interpreted as an invalid date and results in the default value January 1, 2000.

Condition:

Create a project in an English language version of SILworX V2 (or lower).

Then open this project with V8.34. The messages are read and added to the project history as described above.

Workaround:

None.

4.15 Documentation

- Effect:

Cross-references of structure or array elements do not appear in print.

Condition:

Use a structure element or array element and check the cross-reference in the documentation.

Workaround:

None.

4.16 User Management

- Effect:

When restoring a user management archive created with V9, a default user contained in the archive is ignored. The user 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.

4.17 English SILworX

- Effect:

The diagnostic overview for HIQuad X systems shows some messages in German even if the language is set to English.

Condition:

After selecting the language (English), observe the diagnostic overview of one HIQuad X system: some messages are still in German.

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 more details, refer to the safety manual and the chapter on accurately determining the watchdog time.

- 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, 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:

Value changes for VAR_INPUT variables in user-defined function blocks.

In user-defined function blocks, SILworX handles VAR_INPUT variables differently, depending on how the inputs are wired:

- If the inputs are wired with variables of a default data type, the value of the variable 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 content of the online help associated with a user-defined POU.

Condition:

-

Workaround:

Use Windows to display the online help content 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.xx, and *Data type* in V5.xx and higher. When an export file is imported from a version prior to V5.xx, 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:

Hardware login, open the detail view for the module.

Open the diagnostics for this module.

Then close the connection.

After a lost connection, the diagnostics only offers the module login.

The detail view offers a system login.

Workaround:

Once the module login dialog box for the diagnostics has been opened, all online views of this module (diagnostics and module online view) must be closed and then reopened so they can be read again via the system.

- Effect:

For HIMatrix devices prior to F*03 (such as F30 01, F31 02, F35 01, F35 012 or F60 CPU 01), the parameters indicating the status of local forcing (located above the force table) are displayed with regular values as if the information was actually available, but they have no function. In particular, these parameters are *Force State*, *Forced Variables*, *Remaining Force Duration*, and *Force Time Reaction*.

Condition:

-

Workaround:

Refer to the online help for further details.

6 Upgrading from a Previous Version

Project data from previous versions can still be used in V10.58.

No CRC changes occur as long as the minimum configuration version setting remains unchanged for a resource. SILworX ensures compatibility of the CRCs, provided that no changes occur or no new features are used.

Observe the following procedure to upgrade from V2.36 and higher to V10.58:

- A backup should be performed for the project.
- Generate code for all resources prior to conversion. An export and import of the configuration in the version comparator allows potential deviations after the conversion to be detected.
- Open the project in V10.58 and convert it.
- Since the conversion is extensive, check the project integrity after completing the conversion.
- Generate the code in V10.58 to detect potential errors and check if CRCs have changed. A version comparison with the imported configuration will detect this.
- Remove detected errors and re-generate the code to detect CRC changes.
- If no CRC changes are detected, the migration was completed successfully.
- If CRC changes are detected, verify whether they can be accepted.
- If the changes can be accepted, the migration is successfully completed.
- If they cannot be accepted, continue to work with corresponding previous version.

Conversion notes:

- The procedure to convert versions prior to V2.36 is described in the release notes to V2.36.
- For very large projects, the conversion can take several hours.

6.1 References

- SILworX Online Help
- SILworX First Steps Manual, HI 801 103 E
- Communication Manual, HI 801 101 E
- HIPRO-S V2 Manual, HI 800 723 E
- ISOfast Manual, HI 801 079 E
- Modbus V2 Manual, HI 801 475 E
- X-OPC Server Manual, HI 801 480 E

MANUAL
Release Notes


HI 801 499 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 **HIMA** solutions online:

 www.hima.com/en



www.hima.com