User Manual
MasterTool IEC XE
MT8500

MU299609 Rev. O

March 26, 2025

www.altus.com.br

According to current legislation in Brazil, the Consumer Defense Code, we are giving the following information to clients who use our products, regarding personal safety and premises.

The industrial automation equipment, manufactured by Altus, is strong and reliable due to the stringent quality control it is subjected to. However, any electronic industrial control equipment (programmable controllers, numerical commands, etc.) can damage machines or processes controlled by them when there are defective components and/or when a programming or installation error occurs. This can even put human lives at risk. The user should consider the possible consequences of the defects and should provide additional external installations for safety reasons. This concern is higher when in initial commissioning and testing.

The equipment manufactured by Altus does not directly expose the environment to hazards, since they do not issue any kind of pollutant during their use. However, concerning the disposal of equipment, it is important to point out that built-in electronics may contain materials which are harmful to nature when improperly discarded. Therefore, it is recommended that whenever discarding this type of product, it should be forwarded to recycling plants, which guarantee proper waste management.

It is essential to read and understand the product documentation, such as manuals and technical characteristics before its installation or use. The examples and figures presented in this document are solely for illustrative purposes. Due to possible upgrades and improvements that the products may present, Altus assumes no responsibility for the use of these examples and figures in real applications. They should only be used to assist user trainings and improve experience with the products and their features.

Altus warrants its equipment as described in General Conditions of Supply, attached to the commercial proposals.

Altus guarantees that their equipment works in accordance with the clear instructions contained in their manuals and/or technical characteristics, not guaranteeing the success of any particular type of application of the equipment.

Altus does not acknowledge any other guarantee, directly or implied, mainly when end customers are dealing with third-party suppliers. The requests for additional information about the supply, equipment features and/or any other Altus services must be made in writing form. Altus is not responsible for supplying information about its equipment without formal request. These products can use EtherCAT® technology (www.ethercat.org).

COPYRIGHTS

OPEN SOURCE SOFTWARE NOTICE

To obtain the source code under GPL, LGPL, MPL and other open source licenses, that is contained in this product, please contact opensource@altus.com.br. In addition to the source code, all referred license terms, warranty disclaimers and copyright notices may be disclosed under request.

WinPcap

This software uses WinPcap, which is licensed under the WinPcap License. For more information, please refer to the license at https://www.winpcap.org/misc/copyright.htm.

Contents

1. Introduction

Nexto Series is a powerful and complete Programmable Logic Controller (PLC) with unique and innovative features. Due to its flexibility, smart design, enhanced diagnostics capabilities and modular architecture, Nexto can be used for control systems from medium or high-end applications. Due to its compact size and superior performance, Nexto can also be used for small automation systems with time critical requirements.

MasterTool IEC XE is a complete tool for programming, debugging and performing configuration and simulation of user applications. Based on a concept of being integrated, flexible and easy to use, this software provides five programming languages defined by IEC 61131-3 standard: Structured Text (ST), Sequential Function Chart (SFC), Function Block Diagram (FBD), Ladder Diagram (LD) and Continuous Function Chart (CFC). MasterTool IEC XE allows the use of different languages on the same application, providing to the user a powerful way to organize the application and to reuse codes used in previous applications.

This product offers features for every stage of an automation application, starting from initial graphical architecture topology analyses, passing through a programming environment that supports IEC 61131-3 languages and a realistic simulation tool, where the user can verify application's behavior before running in a real system and ending in a complete diagnostics and status visualization interface.

MasterTool IEC XE also offers two different protection schemes as application security features: IP Protection and Secure PLC Login. IP Protection is targeted to protect user's intellectual property, allowing the user to protect the complete project and files by defining a password to access them. This means that these files will be available (both read and write operations) only after unlocking them with the correct password. Secure PLC Login provides a way to protect the user application from any unauthorized access. By enabling this feature, Nexto CPU will request a user and a password before performing any available command in MasterTool IEC XE and Nexto CPU, as stopping, programming and forcing of outputs in the target CPU.

MasterTool IEC XE allows the use of fieldbus interfaces in an easier way than ever seen before. The user does not need any special software to configure fieldbuses anymore, because MasterTool IEC XE covers this requirement providing a unique tool reducing engineering time and making applications more simple.

In order to increase user's productivity, some important features are also available: Module Printing which is a report generation of every module specific parameters and application general settings, Logic Printing which is a report generation of all application code, Enhanced Project Verification which helps user to check several different conditions during programming, like programming syntax, power supply module current consumption, placement rules for Nexto modules, modules parameters and settings, Real Time Debugging which provides useful way to check the application step-by-step, verify variables values or add and remove breakpoints during Nexto CPU programming.

1.1. Documents Related to this Manual

To obtain additional information regarding the MasterTool IEC XE , other documents (manuals and technical features) besides this one, may be accessed. These documents are available in its last version on the site.

1.2. Visual Inspection

Before resuming the installation process, it is advised to carefully visually inspect the equipment, verifying the existence of transport damage. Verify if all parts requested are in perfect shape. In case of damages, inform the transport company or Altus distributor closest to you.

CAUTION

Before taking the modules off the case, it is important to discharge any possible static energy accumulated in the body. For that, touch (with bare hands) on any metallic grounded surface before handling the modules. Such procedure guaranties that the module static energy limits are not exceeded.

It's important to register each received equipment serial number, as well as software revisions, in case they exist. This information is necessary, in case the Altus Technical Support is contacted.

1.3. Technical Support

For Altus Technical Support contact in São Leopoldo, RS, call +55 51 3589-9500. For further information regarding the Altus Technical Support existent on other places, see https://www.altus.com.br/en/ or send an email to altus@altus.com.br.

If the equipment is already installed, you must have the following information at the moment of support requesting:

- The model from the used equipments and the installed system configuration

- The product serial number
- The equipment revision and the executive software version, written on the tag fixed on the product's side
- CPU operation mode information, acquired through MasterTool IEC XE
- The application software content, acquired through MasterTool IEC XE
- Used programmer version

1.4. Warning Messages Used in this Manual

In this manual, the warning messages will be presented in the following formats and meanings:

DANGER

Reports potential hazard that, if not detected, may be harmful to people, materials, environment and production.

CAUTION

Reports configuration, application or installation details that must be taken into consideration to avoid any instance that may cause system failure and consequent impact.

ATTENTION

Identifies configuration, application and installation details aimed at achieving maximum operational performance of the system.

2. Technical Description

2.1. MasterTool IEC XE Versions

MasterTool IEC XE has four distribution versions, each one with an optimized portfolio in accordance with the user's needs.

- **Lite:** free programming software that allows the programming and project loading of up to 320 I/O points
- **Basic:** software that allows the programming and project loading of up to 2048 I/O points
- **Professional:** programming software for all Nexto Series CPUs
- **Advanced:** programming software with tools for large scale applications using half-cluster redundancy

Each one of these versions has unique characteristics, purposes and features for each specific objective.

	Lite	Basic	Professional	Advanced
Free version	Yes	No	No	No
Available languages:	5	5	5	5
Structured Text (ST)	Yes	Yes	Yes	Yes
Sequential Function Chart (SFC)	Yes	Yes	Yes	Yes
Function Block Diagram (FBD)	Yes	Yes	Yes	Yes
Ladder Diagram (LD)	Yes	Yes	Yes	Yes
Continuous Function Chart (CFC)	Yes	Yes	Yes	Yes
Rack expansion support	No	Yes	Yes	Yes
Rack expansion redundancy support	No	No	Yes	Yes
Ethernet expansion support	No	No	Yes	Yes
Ethernet expansion redundancy support	No	No	Yes	Yes
PROFIBUS support	No	Yes	Yes	Yes
PROFIBUS redundancy support	No	No	Yes	Yes
Half-Cluster redundancy support	No	No	No	Yes
Event grouping support	Yes	Yes	Yes	Yes
DNP3 protocol support	No	Yes	Yes	Yes
IEC 60870-5-104 protocol support	Yes	Yes	Yes	Yes
IEC 61850 protocol support	No	Yes	Yes	Yes
Xtorm Ethernet interface redundancy support	No	Yes	Yes	Yes
CPU redundancy support	No	No	No	Yes
Limitation of the number of local I/O points	Yes	Yes	No	No
Maximum number of local I/O points	320	2048	Unlimited	Unlimited

Table 1: Versions Features

Notes:

Continuous Function Chart (CFC): The CFC language has two editors. In the first, all functions are enumerated with a unique execution order. In the second, the user can edit logic groups in individually enumerated pages – therefore it is called *Page Oriented*.

Fieldbus support: Nexto Series architectures use the PROFIBUS DP as the fieldbus.

Maximum number of local I/O points: For this limit, only I/O points present in the CPU rack are considered, not considering I/O points in remote ones. In the case of Advanced and Professional licenses, the limit will be the memory occupation %I and %Q in each CPU model.

	Lite	Basic	Professional	Advanced
CODESYS Control Win V3 x64	Yes	Yes	Yes	Yes
NL717	Yes	Yes	Yes	Yes

	Lite	Basic	Professional	Advanced
XP300	Yes	Yes	Yes	Yes
XP315	Yes	Yes	Yes	Yes
XP325	Yes	Yes	Yes	Yes
XP340	Yes	Yes	Yes	Yes
XP350	Yes	Yes	Yes	Yes
XP351	Yes	Yes	Yes	Yes
NX3003	Yes	Yes	Yes	Yes
NX3004	Yes	Yes	Yes	Yes
NX3005	Yes	Yes	Yes	Yes
NX3008	Yes	Yes	Yes	Yes
NX3010	Yes	Yes	Yes	Yes
NX3020	No	Yes	Yes	Yes
NX3030	No	No	Yes	Yes
NX5100	Yes	Yes	Yes	Yes
NX5101	Yes	Yes	Yes	Yes
HX3040	Yes	Yes	Yes	Yes

Table 2: Supported Products

Note:

CODESYS Control Win V3 x64 support: CODESYS Control Win V3 x64 will be supported directly by CODESYS. Only for DEMO Mode use.

2.2. Languages

MasterTool IEC XE software is available in a few languages. After the installation, the interface assumes the language of Computer Operating System. The language can be changed after installation without the need for resettlement.

2.3. Compatibility with Other Product

Versions of MasterTool IEC XE are not compatible with all versions of controllers. To know which version is compatible,the technical datasheet document of each controller should be consulted.

2.4. Minimum and Recommended Requirements

MasterTool IEC XE presents as minimum and recommended requirements for its installation and utilization the following specs:

	MasterTool IEC XE
Platform	PC with operating system: **Until version 3.05:** Windows XP® (32 bits), Windows Vista® (32 bits), Windows 7 SP1® (32 bits or 64 bits) or Windows 8.1® (64 bits) **From version 3.10 until version 3.23:** Windows 7 SP1® (32 bits or 64 bits) or Windows 8.1® (64 bits) **From version 3.30 until version 3.35:** Windows 7 SP1® (32 bits or 64 bits), Windows 8.1® (64 bits) or Windows 10® (64 bits)

	MasterTool IEC XE
	Version 3.40: Windows 8.1® (64 bits),Windows 10® (64 bits) or Windows 11® (64 bits) **From version 3.50:** Windows 10® (64 bits) or Windows 11® (64 bits)
Processor	2.5 GHz (recommendable)
Disk Space	2 Gbyte (minimum), 12 Gbytes (recommended)
RAM	4 Gbytes (minimum), 16 Gbytes (recommended)
Resolution	1024 x 768 (recommended)
Language	Any language

Table 3: Minimum and Recommended Requirements for Installation and Operation

Note:

Requirements: As a rule, PCs that fill the minimum requirements can be used for non-redundant applications. Redundant applications should use PCs that have at least the recommended settings.

2.5. Data for Purchase

2.5.1. Included Items

The MasterTool IEC XE software is sold as a service, with the contract and the respective license sent in digital format to the customer. Get in touch with the Altus commercial department if physical media is required with the product.

2.5.2. Product Code

The following codes shall be used for product purchase:

Code	Description
MT8500 Lite	MT8500 Lite
MT8500 /BASIC/S	MT8500 Basic
MT8500 /PRO/S	MT8500 Professional
MT8500 /ADV/S	MT8500 Advanced

Table 4: Product Code

3. Concepts and Basic Components

3.1. Introduction

MasterTool IEC XE is a device-independent PLC software programming. Matching the IEC 61131-3 standard it supports all standard programming languages.

3.2. Basic Concepts

Regard the following basic concepts determining programming with MasterTool IEC XE :

- **Object Orientation:** The mind of object orientation is not only reflected by the availability of appropriate programming elements and features but also in the structure and version handling of MasterTool IEC XE and in the project organization.
- **Component-based structure of the programming system:** The functionality available in the user interface (editors, menus etc.) depends on the currently used components defined in a profile. There are system components, which are essential and optional components.
- **Project organization is also determined by the mind of object orientation:** A MasterTool IEC XE project contains a PLC program composed of various programming objects and it contains definitions of the *resources* which are needed to run instances of the program (application) on defined target systems (devices, PLCs).So there are two main types of objects in a project:
  - **Programming objects:** Programming objects (POUS) which can be instantiated in the entire project, i.e. for all applications defined in the project, must be managed in the POUs window. These are programs, functions, function blocks, methods, interfaces, actions, data type definitions etc. The instantiating is done by calling a program POU by an application-assigned task. Programming objects, which are managed in the devices window, i.e. which are directly assigned to an application, cannot only be instantiated by another application inserted below.
  - **Resource objects:** These are device objects, applications, task configurations, and recipe managers etc., which are managed in the *device tree* or in the graphic editor (depending on the device type). When inserting objects in the devices tree, the hardware to be controlled must be mapped according to certain rules.
- Code generation by integrated compilers and use of machine code results in short execution times.
- **Data transfer to the controller device:** The data transfer between MasterTool IEC XE and the device is done via a gateway (component) and a runtime system.

3.3. Advanced Functionalities

In the following, the user can see the advanced functionalities available on MasterTool IEC XE .

3.3.1. Object Orientation in Programming and Project Structure

Extensions to function blocks: Properties, Methods, Inheritance, Method invocation.

Applications are instances of independent programming objects.

3.3.2. Special Data Types

- ANY_TYPE
- UNION
- LTIME
- References
- Enumerations: base data type can be specified
- DI: DINT := DINT#16#FFFFFFFF

3.3.3. Operators and Special Variables

- Scope operators: extended namespaces
- Function pointers: replacing the INSTANCE_OF operator
- Init Method: replacing the INI operator
- Exit Method
- Output variables in function and method calls
- VAR_TEMP/VAR_STAT/VAR_RETAIN/ VAR_PERSISTENT...
- Arbitrary expressions for variable initialization
- Assignments expression
- Index access with pointers and strings

3.3.4. New User Management and Access Rights Concept

- User accounts, user groups, group-specific rights for access and actions on particular objects.

3.3.5. Features in Editors

- ST-Editor: Editing resources, Line break, Autocomplete, Inline Monitoring, and Inline set/reset-assignment.
- FBD and LD are reversible and programmable in a combined editor.
- LD/FBD editor: Main output in boxes with multiple outputs can be changed
- LD/FBD editor: no automatic update of box parameters
- LD/FBD editor: branches and *networks within networks*
- SFC editor: only one step type, macros, multiple selections of independent elements, no syntactic check during editing, automatic declaration of flag variables

3.3.6. Library Versions

- Multiple versions can be used in the same project using the context resource
- Installation in repositories, automatic update and debugging

3.3.7. Additional Functionalities

- PLC Configuration and Task Configuration integrated in devices tree
- UNICODE support
- Single-line comments
- Watchdog
- Multiple selection in the project object tree
- Online help integrated in the user interface
- Conditional compilation
- Conditional breakpoints
- Debugging: step to cursor, back to previous caller
- Field bus driver matching IEC 61131-3
- Symbolic and PLC configuration available in application
- Free memory allocation of code and data
- Precompile notes concerning syntax errors

3.4. Project Profiles

A MasterTool IEC XE project profile is a set of rules, common characteristics and patterns used in the development of an industrial automation solution. It is a profile that influences the form of implementation of the application. With the diversity of types of applications supported by Nexto Series Runtime System (RTS), to follow a profile is a way to reduce the complexity in programming.

Applications can be created according to one of the following profiles:

- Single
- Basic
- Normal
- Expert
- Custom
- Machine Profile

MasterTool IEC XE provides several templates compatible with each profile defined for the RTS. When the user selects a template as a model in project creation, the new application will be developed according to a certain profile, adopting the rules, characteristics and standards defined by the profile associated to the template. For further information about each of these profiles consult the specific manual for each controller.

Note: Throughout the project profile, are named some types of tasks which are described in the Task Configuration section.

3.5. Project

A project contains the POU objects which make up a PLC program, as well as the definitions of resource objects necessary to run one or several instances of the program (application) on certain target systems (PLCs, devices). POU objects are managed in the POUs view window, device-specific resource objects are managed in the Devices view window.

A project is saved in a file <project name>.project.

Note: The look and the properties of the user interface are defined and stored in MasterTool IEC XE , not in the project.

3.5.1. Project Template

The basic configuration of a new project (menu structure, predefined objects) is determined by the currently used project template. The template is chosen when creating a new project file by command *New Project*. For further information see Start MasterTool IEC XE .

3.6. POUs Window

In the *POUs* window, the user can add POUs, external files, etc. Besides that, it shows configurations objects and project information.

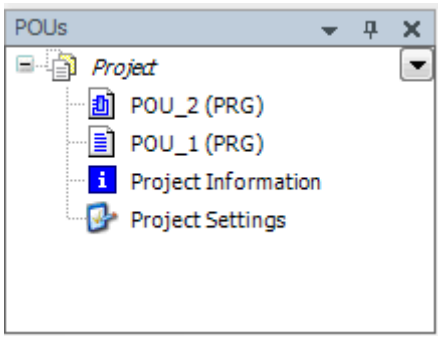

Figure 1: POUs Window

3.6.1. Project Settings and Project Information

The *Project Settings* and *Project Information*, per default are objects in the POUs view.

The *Project Settings* dialogs provide the possibility to define various settings for the current project, e.g. security, page configurations, etc.

The *Project Information* dialogs can be used to edit or view project specific information like e.g. file data, statistics on the objects, author name etc.

For a detailed description see Project Menu.

3.6.2. External File

Symbol: 

Any external file (with any extension) can be added to the POUs view of a project via the *Add Object* command. In the *Add object* dialog choose object type *External File*.

Press button  for getting the dialog for browsing for a file, the path of which will be entered to the field below *File path*. In the field below *Name* automatically the name of the chosen file will be entered without extension. The user can edit this field to define another name for the file under which it should be handled within the project.

Select one of the options:

As long as the external file is available as defined, the defined update options will effect accordingly. Otherwise, just the file version stored in the project will be available.

- **Remember the link:** The file will be available in the project only if it is available in the defined link path.
- **Remember the link and embed into project:** A copy of the file will be stored internally in the project but also the link to the external file will be remembered. When the external file changes, then: If the external file is linked to the project, the user can additionally select one of the options.
  - **Reload the file automatically:** The file will be updated within the project as soon as it has been changed externally.
  - **Prompt whether to reload the file:** A dialog will pop up as soon as the file has been changed externally. The user can decide whether the file should be updated also within the project.
  - **Do nothing:** The file will remain unchanged within the project, even when it is changed externally.
- **Embed into project:** Just a copy of the file will be stored in the project. There will be no further connection to the external file

In the dialog there is the button *Display File Properties...*. This button opens the standard dialog for the properties of a file, which also appears when the user selects the file object in the POUs window and use command *Properties*.

The dialog contains a tab *External file* where the properties, which have been set in the *Add Object* dialog, can be viewed and modified.

3.7. Device, Device Tree

In the *Devices* window (*Device tree*) the hardware can be mapped on which the application is to run.

Each *device* object represents a specific (target) hardware object. Examples: controller, field bus node, bus coupler, drive, I/O-module, monitor.

Each device is defined by a device description and must be installed on the local system in order to be available for inserting in the devices tree (Figure 2). The device description file defines the properties of a device concerning configurability, programmability and possible connections to other devices.

In the device tree however not only device objects are managed, but all objects which are needed to run an application on a device (controller, PLC); thus also the *Application* objects as well as *Task Configuration* and *Task* objects. But also, pure programming objects like particular POUs, Global Variable lists and Library Manager can - instead of being managed as project-globally instantiable units in the POUs window - be managed ONLY in the device tree and in this case are only available for exactly *their* application or *child applications* of their application.

See in the following some Generals on the device tree and information on the installation and the arranging of the objects.

3.7.1. Generals

In the *Devices* window (*Device tree*) the hardware can be mapped on which the application is to run.

Each *device* object represents a specific (target) hardware object. Examples: controller, field bus node, bus coupler, drive, I/O-module, monitor.

Each device is defined by a device description and must be installed on the local system in order to be available for inserting in the devices tree (Figure 2). The device description file defines the properties of a device concerning configurability, programmability and possible connections to other devices.

In the device tree however not only device objects are managed, but all objects which are needed to run an application on a device (controller, PLC); thus also the *Application* objects as well as *Task Configuration* and *Task* objects. But also, pure programming objects like particular POUs, Global Variable lists and Library Manager can - instead of being managed as project-globally instantiable units in the POUs window - be managed ONLY in the device tree and in this case are only available for exactly *their* application or *child applications* of their application.

See in the following some Generals on the device tree and information on the installation and the arranging of the objects.

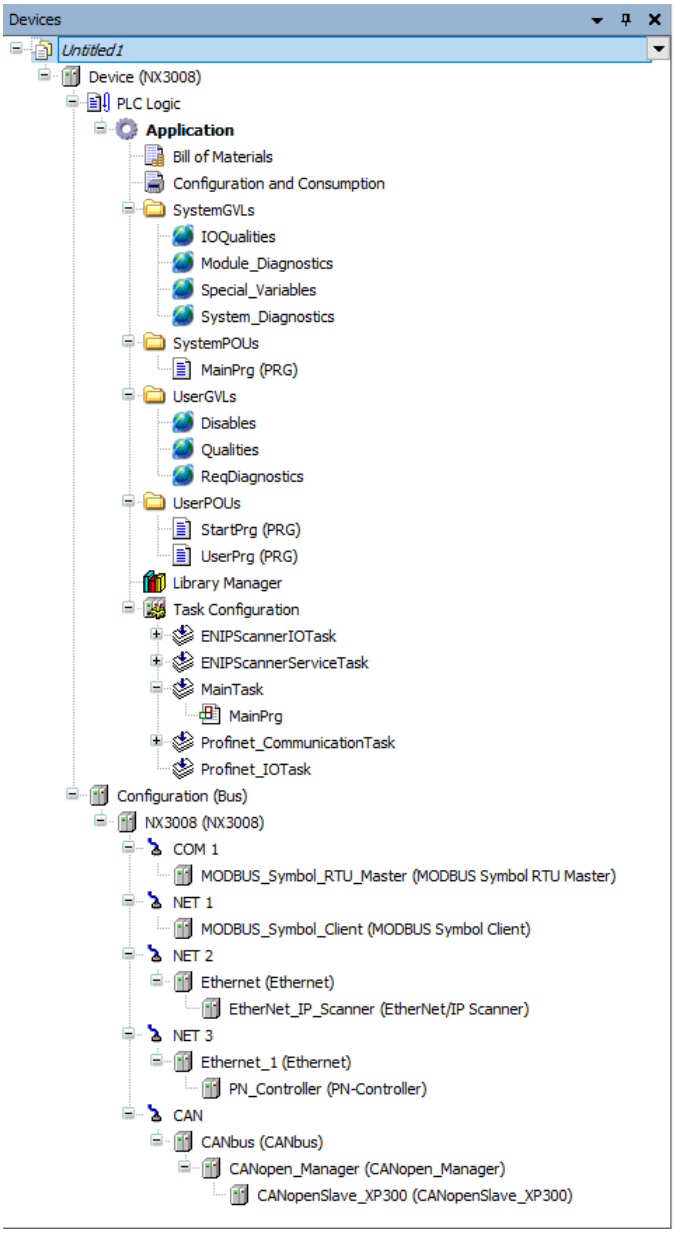

Figure 2: Device Tree

- The root node of the tree always is a symbolic node entry : <projectname>.
- Each entry in the device tree shows the symbol, the symbolic name (editable) and the device type.
- A device is programmable or just parameterizable. The type of the device determines the possible position within the resources tree and also which further resources can be inserted below the device. Programmable devices are indicated by an additional ( ) *Plc Logic* node inserted automatically below the device entry. Below this node the objects needed for programming the device (applications, text lists etc.), as well as functional objects like e.g. a Parameter Manager, can be inserted. Pure parameterizable devices cannot get assigned such programming objects, however the values of the device parameters might be edited in the parameter dialog of the device editor. Regard that the programmability of a device is a property which can change (device description) without the need of reinserting the device.
- The configuration of a device concerning communication, parameters, I/O Mapping is done in the Device dialog (Device Editor), which can be opened by a double-click on the device entry (see Device Editor for a description).
- In online mode an icon at the beginning of a device entry indicates whether the device currently is connected or not connected.

Symbol	Status
	The PLC is connected, the application is running, the device is in operation, and data is being exchanged.
	The PLC is connected and in STOP.
	The device is not exchanging data; bus error, no configuration, or simulation mode.
	The device is running in trial mode for 30 minutes. After this time has elapsed, the trial mode will expire and the fieldbus will terminate the data exchange.
	The device is configured, but not fully operational. No data is exchanged. Example case: CANopen devices when booting and in preoperative mode.
	Redundancy mode is active. The fieldbus master is not sending any data because another master is active.
	The device description could not be found in the device repository.
	The device itself is running, but a child device is not running or it has a diagnostic message. The child device is not visible due to a collapsed device tree.
	Gray exclamation mark: The device itself is running, but a subordinate device is not running or has a diagnostic message. A diagnosis was pending. The cause of the error no longer exists. This symbol can appear in connection with various other symbols in this list.
	Red exclamation mark: The device is not running or a diagnosis is pending. The error cause still exists. This symbol can appear in connection with various other symbols in this list.

Table 5: Device Status in Online Mode

3.7.2. Installing of Devices on the Local System

- Most of the required devices for already MasterTool IEC XE use are already installed automatically. Installation and uninstalling of devices can be done in the *Device Repository* dialog (see Device Repository). The installation bases on device description files in xml-format. The default extension for a valid device description file is *.devdesc.xml. However also bus-specific description files like *.gsd-files (PROFIBUS), can be installed via the *Device Repository* dialog. The specific MasterTool IEC XE previous version PLC configuration files *.cfg can be used if an appropriate additional info-file *.info.cfg, describing the respective set of configuration files, is provided.

3.7.3. Arranging and Configuring Objects in the Devices Tree - Rules

- To add an object use command *Add Device*. The device types that can be inserted depend on the currently selected object within the device tree. For example: modules for a DP Profibus slave cannot be inserted without having inserted

an appropriate slave device before; no applications can be inserted below non-programmable devices. Further on only devices correctly installed on the local system and matching the current position in the tree will be available for insertion.

- Re-positioning of objects is possible via the standard commands of clipboard (Cut, Copy, Paste, Delete) or by drawing the selected object with the mouse while the mouse-button <CTRL> (for copying) is pressed.
- Only *device* objects can be positioned on the level directly below the root node <projectname>. If currently no entry is selected, e.g. if you click with the mouse in the empty space of the devices view, this equals to having selected the root node entry.
- A device will be inserted as a node in the tree. If defined in the device description file, automatically sub-nodes might be inserted. A sub-node again might be a programmable device.
- Below a *device* object further devices might be inserted, if those are installed on the local system and thus available in the *Add Object* and *Add Device* dialogs. The sorting of device objects within the tree from up to down: On a particular tree level first always the programmable devices (Plc Logic) are arranged, followed by any further devices, each sorted alphabetically.
- As an assistance for setting up the PLC configuration in the device tree a scan functionality is provided by the standard device editors. The current hardware structure can be read and displayed in a dialog, allowing the user to direct take over the desired modules to the project's device tree in the project.

3.8. Application

An *application* is a set of objects which are needed for running a particular instance of the PLC program on a certain hardware device (PLC, controller). For this purpose *independent* objects, managed in the POUs view, are instantiated and assigned to a device in the *Devices* view. This meets the mind of object-orientated programming. However also purely application-specific POUs can be used.

An application is represented by an application object ( ) in the devices tree , insertable below a Plc Logic (programmable) device node. Below an application entry the objects defining the applications *resource set* are to be inserted.

The standard application, *Application* is created with new projects from Standard Project and is added to the device tree below the *Device* and *PLC Logic* items.

An essential part of each application is the *Task Configuration* controlling the run of a program (POU instances or application-specific POUs). Additionally it might have assigned resource objects like *Global Variables Lists*, *Libraries*, etc., which - in contrast to those managed in the *POUs* window - only can be used by the particular application and its *children*.

When going to log in with an application on a target device (PLC or simulation target), it will be checked which applications currently are on the PLC and whether the application parameters on the PLC are matching those in the project configuration. Appropriate messages will indicate mismatches. See the description of the Login command for further details.

3.9. Task Configuration

The *Task Configuration* ( ) defines one or several tasks for controlling the processing of an application program.

It is an essential resource object for an application and it is inserted automatically when you create a new project from Standard Project. A task can call an application-specific program POU, which is only available in the device tree below the application, as well as a program which is managed in the POUs window. In the latter case the project-globally available program will be instantiated by the application.

A task configuration can be edited in the task editor, the available options being target-specific. In online mode the task editor provides a monitoring view giving information on cycles, cycle times and task status.

3.9.1. Important Notes for Multitasking Systems

On some systems real preemptive multitasking is realized. In this case the following must be regarded:

- All tasks share one process map. Reason: An own process map for each task would charge the performance. Therefore, the process map always can only be consistent to one task. Hence the user, when creating a project, explicitly must take care that in case of conflicts the input data will be copied to a save area, the same problems have to be regarded for the outputs. For example, modules of the *SysSem* library could be used to solve the synchronization problems.
- Also when accessing other global objects (global variables, modules), consistency problems might occur as soon as the size of the objects exceeds the data width of the processor (structures or arrays forming a logical unit). In addition, the modules of the *SysSem* library might be used to solve the problems.

3.10. Code Generation and Online Change

3.10.1. Code Generation and Compile Information

Machine code will not be generated until the application project gets downloaded to the target device (PLC or simulation target). At each download the compile information, containing the code and a reference ID of the loaded application, will be stored in the project directory in a file *<project name>.<device name>.<application ID>.compileinfo*. The compile info will be deleted when the *Clean* and *Clean all* commands are performed.

3.10.2. Online Change

If the application project currently running on the controller has been changed in the programming system since it has been downloaded last, just the modified objects of the project will be loaded to the controller while the program keeps running there.

ATTENTION

Online Change modifies the running application program and does not affect a restart process. Make sure that the new application code nevertheless will affect the desired behavior of the system. Depending on the controlled system, damages to machines and parts could result, or even health and life of persons could be endangered.

Notes:

- When an online change is done, the application-specific initializations (homing etc.) will not be executed because the machine keeps its state. For this reason the new program code might not be able to work as desired.
- Pointer variables keep their values form the last cycle. If there is a pointer on a variable, which has changed its size due to an online change, the value will not be correct any longer. Make sure that pointer variables get re-assigned in each cycle.

There are two ways to perform an Online Change:

1. As soon as you try to log in again with a modified application (checked via the *compileinfo*, which has been stored in the project folder during the last download), you will get asked whether you want to do an online change, a download or login without changing.

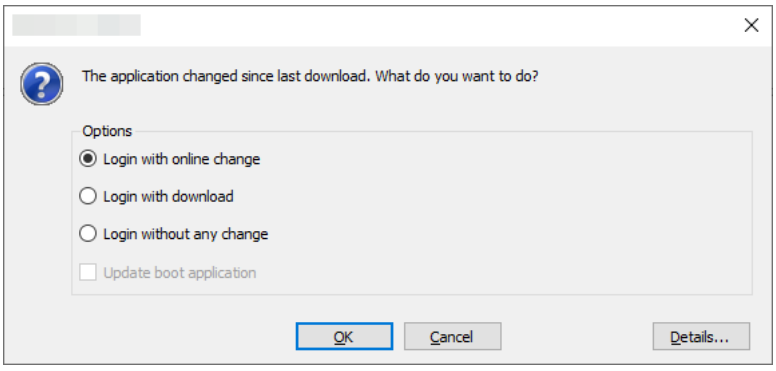

Figure 3: Login Dialog

- **Login with online change:** This option is selected per default. Therefore, if you confirm the dialog with OK, the modifications will be loaded and immediately shown in the online view (monitoring) of the respective object(s).
- **Login with download:** Activate this option if the application project should be compiled and loaded completely.
- **Login without any change:** Activate this option in order to keep the program running on the controller unchanged. Afterwards an explicit download might be done, thus loading the complete application project, or at the next re-login you will be asked again whether an online change should be done.

ATTENTION

In case of projects with *Visualization*, the *Disk is Full* message may appear when you download the application, this means that memory allocated for visualization files is full. In this case, this funcionality may not work correctly and it is recommended to resolve the problem, to execute a *Reset Origin* and try to download again.

Via the *Details...* button in the login dialog you can get some information (Project name, Last modification, IDE version, Author, Description) on the currently concerned application within the IDE (integrated development version = programming system) in comparison to that currently available on the PLC:

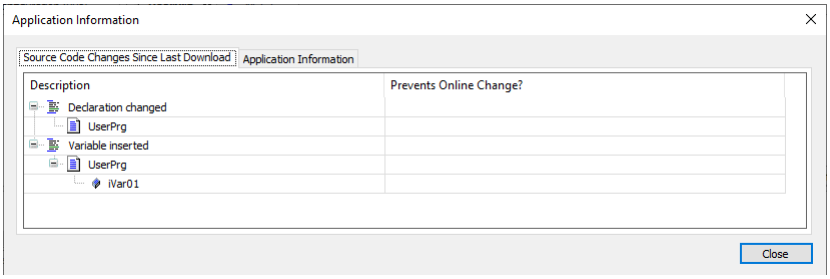

Figure 4: Source Code Changes Since Last Download Dialog

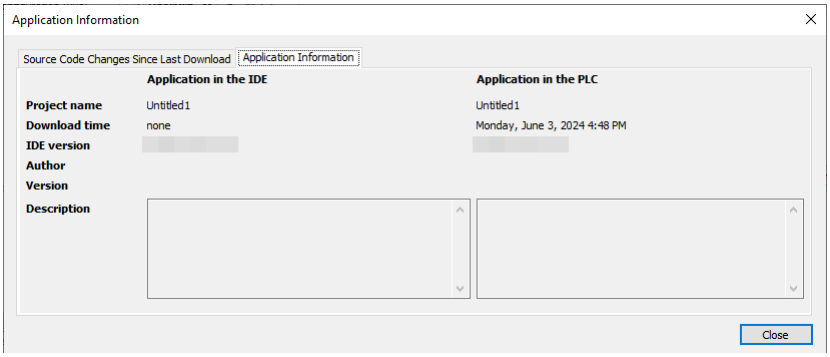

Figure 5: Application Information Dialog

If other applications are currently available on the PLC, you will get additional dialog boxes on how to handle the situation. For a description please see Login.

2. By using the command *Online Change* <application> (*Online* menu) you can explicitly perform an online change on a particular application.

Regard that online change of a modified project is no longer possible after a *Clean* operation (*Clean all*, *Clean*). In this case, the information on which objects have been changed since the last download will be deleted. Thus, only the complete project can be downloaded.

Consider the following before going to do an online change:

Is the changed code without any errors?

Application specific initializations (reference run etc.) will not be processed, because the machine is keeping its state.

Can the new program code really do without such initializations?

Pointer variables keep their value from the last cycle. If the user points on a variable, which now has changed its size, the value will not be correct any longer. For this reason, the user should re-assign pointer variables in each cycle.

If the active step in a function chart gets removed, the chart will remain inactive.

Note: Online changes may not be applied when you change device parameters such as bus modules, PROFIBUS parameters and MODBUS mappings. The same is true if you add or remove devices and also for tasks configuration.

3.10.3. Boot Application (Boot Project)

A boot application is the project which will be started automatically when the controller gets started (*booted*). For this purpose, the project must be available on the PLC in a file *<project name>.app*. This file can be created in offline or online mode by command *Create boot application* (*Online* menu).

At each successful download, automatically the active application will be stored in a file *<application >.app* in the target system folder, thus available as boot application. Command *Create boot application*' allows also in offline mode to save the boot application to a file.

3.10.4. Projects Download/Login Method Without Project Differences

In order to ensure that the user will not have problems on sending and logging same projects in the CPUs running from different stations, it can be performed the following steps after sending a project:

- In the *Additional files* dialog (*Project*, *Project Settings*, *Source Download* menu and *Additional files..* button) select the option *Download information files*.
- Close all dialogs by clicking *OK*.
- Run the command *Source download* (File menu).
- In the *Select Device* dialog that opens, choose the CPU on which the project was sent.
- Close the dialog by clicking *OK*, wait for the download of the project.

To login on running CPUs without generating changes on project from different stations, you must open a *Project Archive* generated from the original project and execute the *Login* command. In the absence thereof may be made of the following steps:

- Run the command *Source upload...* (File menu).
- In the *Select Device* dialog that opens, choose the CPU on which the project was sent.
- Close the dialog by clicking *OK*, wait for the loading of the project.
- In the *Project Archive* dialog, which opens at the end of the loading process, choose the folder to extract and click on *Extract*.
- The project will open and *Login* command can be executed in the corresponding CPU.

For further information see: File Menu and Online Menu.

3.11. Monitoring

In online mode there are various possibilities to display the current values of the watch expressions of an object on the PLC:

- Inline monitoring in the implementation editor of an object. For details see the description of the respective editor.
- Online view of the declaration editor of an object. For details see the description of the Declaration Editor.
- Object-independent watch lists. For details see the description of the watch views.
- *Trace* sampling. Recording and display of variable values from the PLC. For details see the description of the Trace functionality.

Note: In online mode there is a limitation of 25000 entry variables monitorable in POUs edited with the ST editor, the user will be alerted when the limit is exceeded with a build error.

3.12. Debugging

To evaluate programming errors you can use the MasterTool IEC XE debugging functionality in online mode. In this context regard the possibility to check an application in simulation mode, i.e. without the need of connecting to a real hardware target device.

Breakpoints can be set at certain positions to force an execution break. Certain conditions, such as which tasks are concerned and in which cycles the breakpoint should be effective, can be set for each breakpoint. Stepping functions are available to get a program executed in controlled steps. At each break the current values of the variables can be examined. A call stack can be viewed for the currently reached step position.

3.12.1. Breakpoints

A breakpoint, set in an application program, will cause a break during the execution of the program. The possible breakpoint positions depend on the editor. In each case, there is a breakpoint at the end of a POU. See Breakpoints for a description of the commands concerning breakpoints.

3.12.2. Stepping

Stepping allows a controlled execution of an application program, e.g. for debugging purposes. Basically you step from one instruction to the next one by repeated use of the key <F10>, but also the user can step over POUs which are called.

- The next statement to be executed can be explicitly defined (Set next Statement).
- The next execution break can be defined simply by placing the cursor there (Run to Cursor).
- Step Out steps back to the previous caller.

See Breakpoints for a description of the stepping commands. Symbols used in text editors: (➪). Current step position, indicated by a yellow arrow before the respective line and a yellow shadow behind the concerned operation.

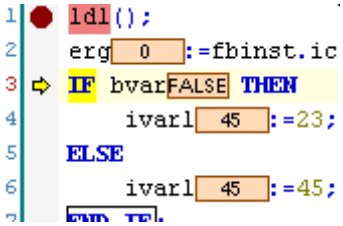

Figure 6: Step Into

3.13. Printing

The currently active editor view can be printed by using the *Print* commands.

Regard also the possibility to create a *documentation* of several or all objects of the project, with a defined layout and a table of contents.

3.14. Security

3.14.1. Project

The access control for projects, particular objects and the right to perform certain actions in a project can be configured and managed via dialogs of the *Project Settings* and *Object Properties*.

A project basically can be protected by an encrypted password, see Project Settings.

Access rights concerning objects always are assigned to particular user groups, not to single users. For each object a list of allowed actions can be defined for each user group. Each particular user however can get an own password.

See User Management and Access Right Management for an overview on user group and access right management for a project.

3.14.2. PLC

Depending on the device there might be an access control related to objects and files in the PLC. See Users and Groups and Access Rights to get information on the respective user and possibilities of management rights in the device Configuration editor

4. Quick Start

The main goal of this chapter is to indicate the basic steps for the Nexto Series CPUs programming. Following this chapter, the user will be able to walk the first steps before start a PLC programming.

- Create and run projects
- Uninstallation, update and repairs
- Getting help

4.1. Start MasterTool IEC XE

From the Start menu on your PC choose MasterTool IEC XE option.

Alternatively, the user can start via the MasterTool IEC XE icon which will be available on the desktop after installation.

Initially the user must create a new MasterTool IEC XE project from the *File* menu followed by *New Project...*, as shown on figure below:

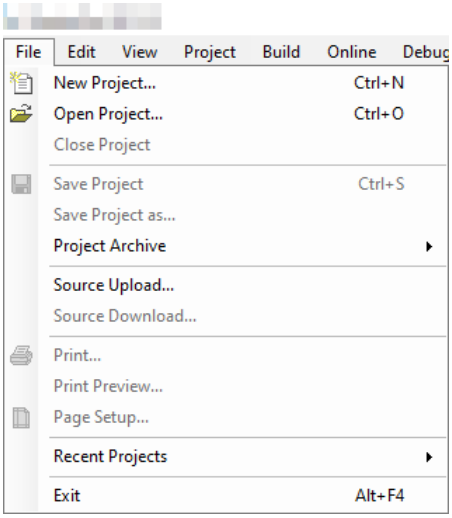

Figure 7: New Project

Later, a window will be presented to the user, requesting the project type selection and, following, the name and path to store the project in the computer. Click on *OK* to proceed or *Cancel* to cancel.

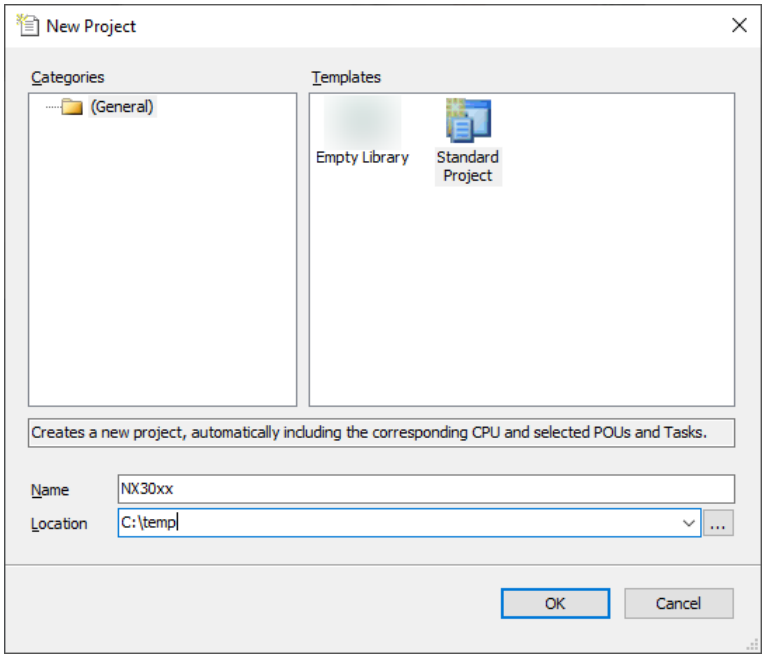

Figure 8: Project Classification

Choosing the Standard Project, a wizard for project creation will be opened, where the user must select the desired CPU and the basic hardware modules composing the bus (considering the model of the rack, the power supply and the redundancy configuration). In this case, the NX3008 CPU will be used with a NX9000 rack. This rack does'nt support redundancy, but racks like NX9002 have redundancy configurations of *Without Redundancy* and *With Redundancy*.

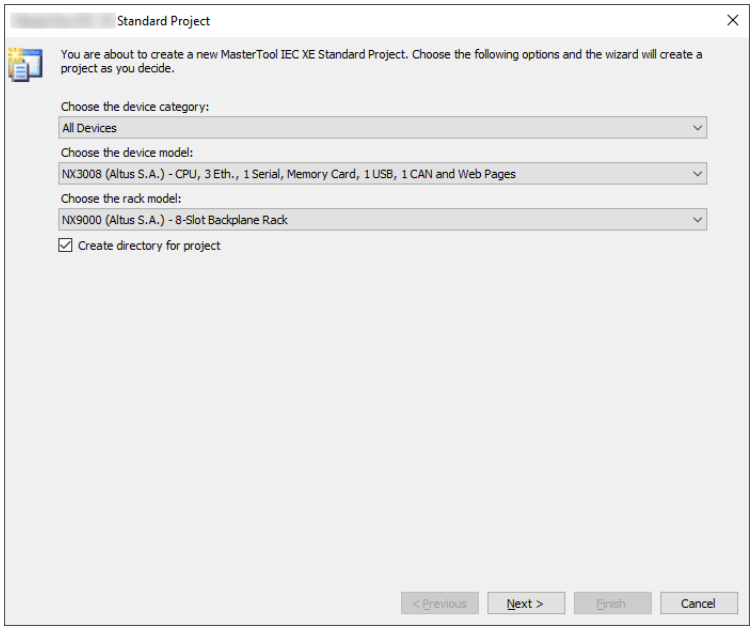

Figure 9: Hardware Modules

Now you can configure the operating mode of the NETs. These modes can be *Single Mode*, *Redundant Mode*, *Switch Mode* and *Disabled* aswell. The only NET that cannot be disabled is the *NET 1*.

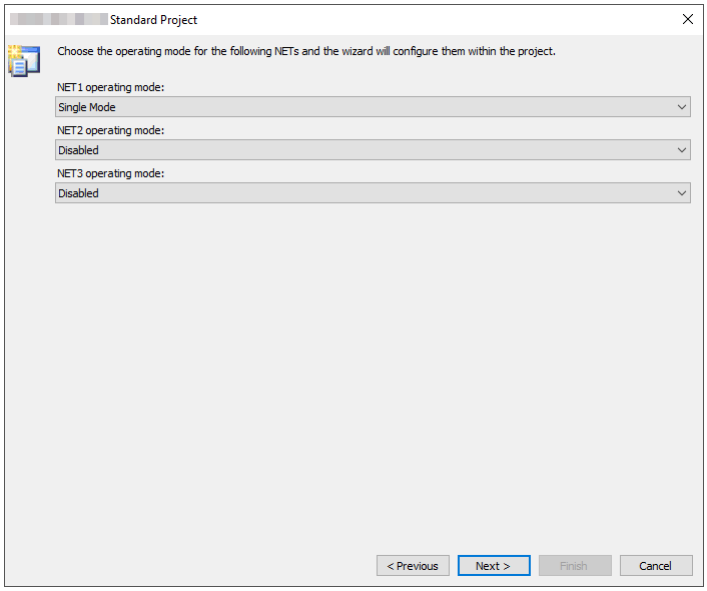

Figure 10: Wizard NET's Configuration

The next step is to choose the network options. In this page it's possible to choose the quantity of PROFIBUS and Ethernet networks. It's also possible to define if these communication networks are going to be simple or redundant. In the case of PROFIBUS, the networks can be *Single Network* or *Redundant Network*. In case of Ethernet, they can be *Single Network with failure mode disabled* or *Redudant Network with Failure Mode Disabled*.

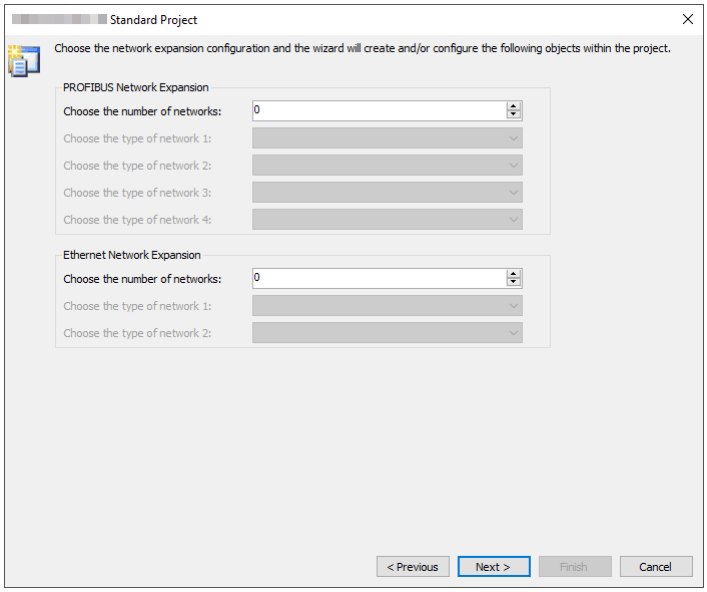

Figure 11: Network Options

The next page allows the definition of the number of I/O points that are going to be automatically created with the project. It's not necessary to know the products codes, simply insert the quantity of points of the application, as the Wizard calculate the quantity of modules needed and adds them. In this example, no I/O points are being created.

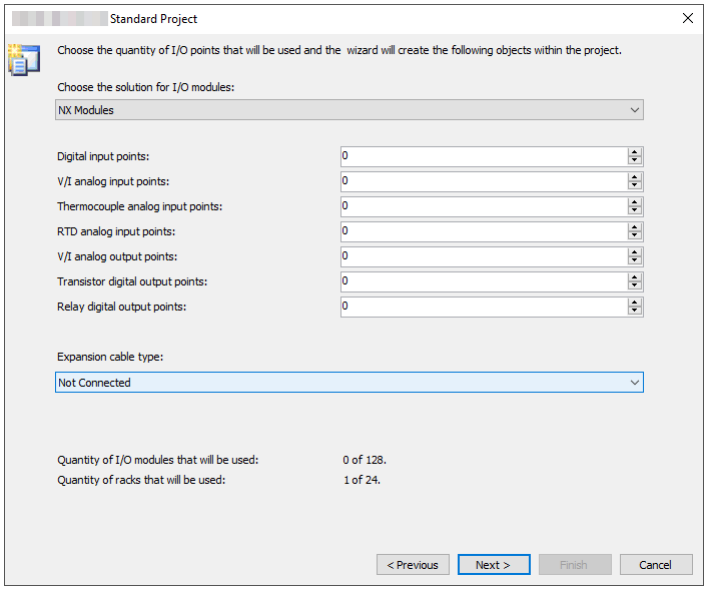

Figure 12: I/O Points Configuration

Following, the user must select the Project profile and the standard language for POUs (programs). In this case, the *Machine Profile* profile and ST language will be used. Click on *Next* to proceed or *Cancel* to cancel.

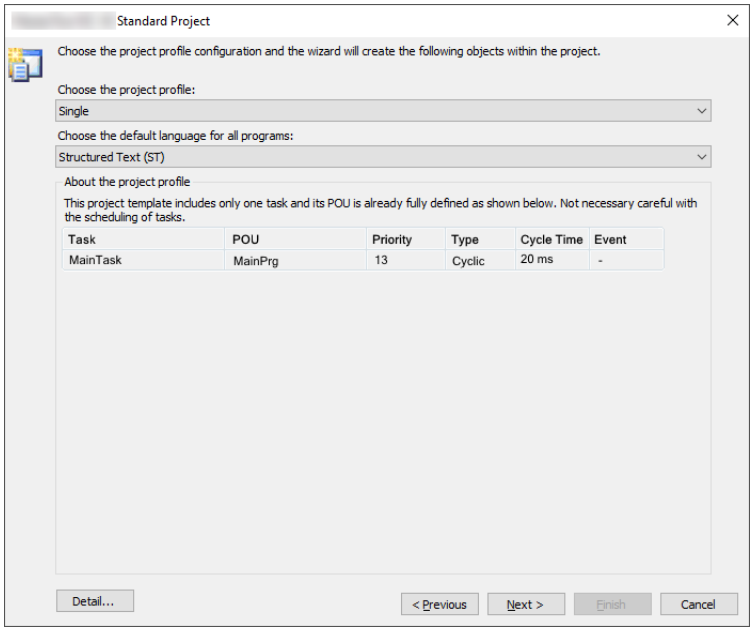

Figure 13: Profile Selection

The next screen defines the POU language created by the selected profile. As the profile is *Single*, there is only three POUs, but MainPrg can't be changed from ST language. The other POUs are UserPrg and StartPrg, they can be their languages changed. Click on *Previous* to return to the last screen, *Finish* to end or *Cancel* to cancel.

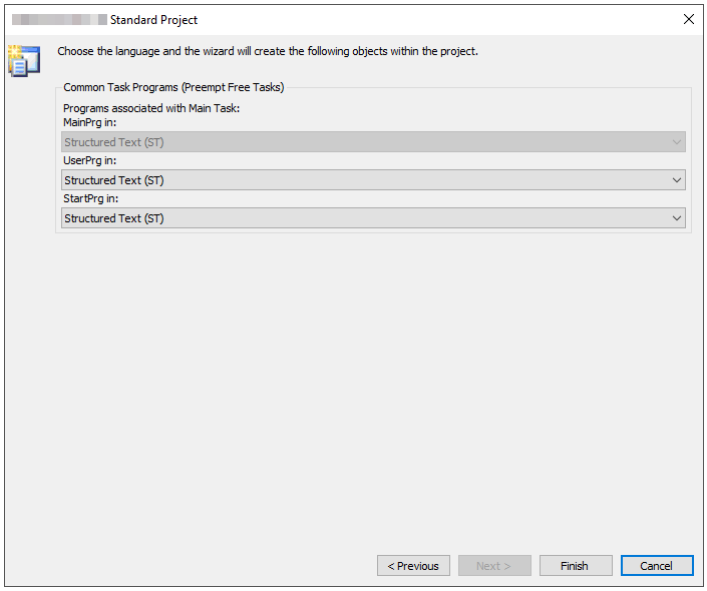

Figure 14: Programming Language

When the *Finish* button is pressed, the MasterTool IEC XE will start the project development environment creation. This procedure may take a few seconds.

4.2. Adding Modules

By default, the CPU and the hardware modules selected at the moment of project creation are already inserted in the system configuration. The user must include the other modules if necessary, then.

In case the tab *Product Library* is not available on the MasterTool IEC XE screen, it must be included through menu *View*, clicking on the item *Product Library*, as shown on figure below:

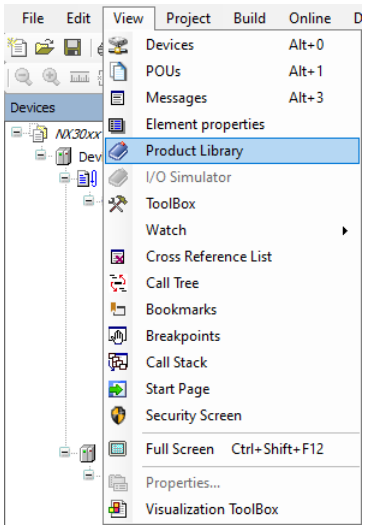

Figure 15: Library Visualization

Then, the module to be inserted in the project must be selected and dragged to the bus configuration area pressing the mouse left button.

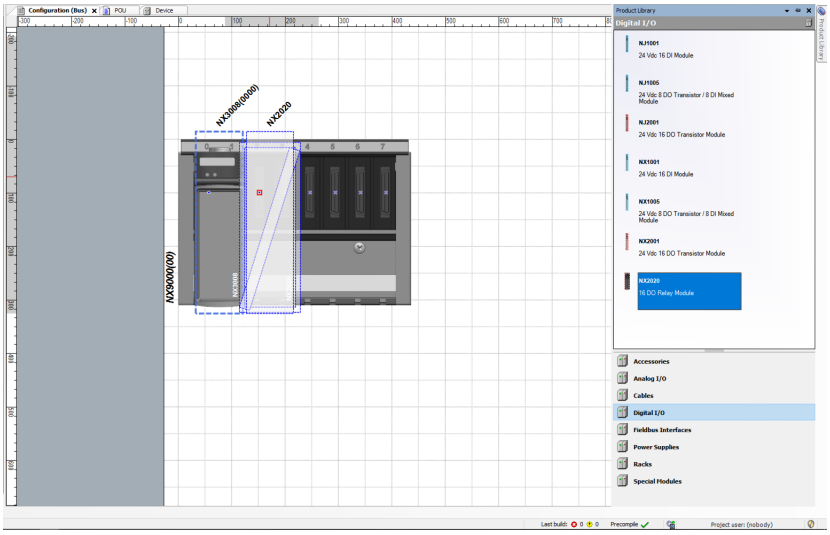

Figure 16: Adding Modules

4.3. Creating POUs

A POU (Program Organization Unit) is a subdivision from the applicative program, which can be written in any language available in the MasterTool IEC XE software.

With the project creation through a selected profile, some POUs are already created, but the user can create as many as he wants, limited by the program memory size.

To add a new POU, you should right-click on Application (the default name created for the application) and select *Add Object* and *POU...*, as shown in the figure below:

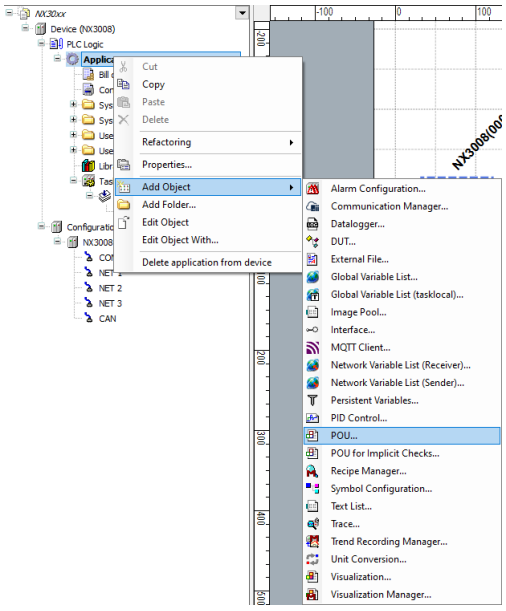

Figure 17: Inserting POUs

A configuration window will appear on the screen, where the user must put the POU name and select the language type to be implemented. Then, the button *Add* must be clicked.

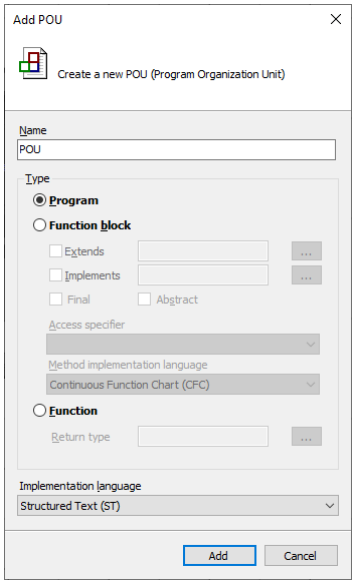

Figure 18: Classification

To edit a POU, select the tab with the corresponding name and start developing the application in the selected language. The same procedure applies to POUs created automatically by the project profile.

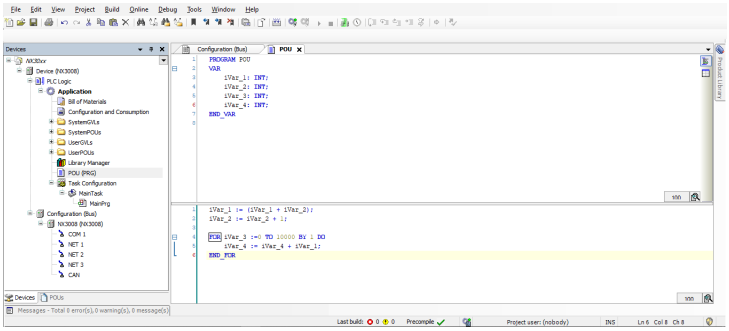

Figure 19: POU Editing

4.4. Creating Tasks

In order for a POU to be executed, it must be associated with a task. This scaling mechanism, called Task, is very useful for real-time systems defined by the periodic execution or upon request of an event (change of state of any Boolean variable). The tasks control the program execution at different speeds, according to the characteristics of the application. The need to execute programs at different rates has the goal of meeting the demand for process under control response time and optimizing the CPU processing capacity. The controllers using tasks are called multitask.

It will only be allowed the creation of new tasks when the selected project profile is the Custom, as in the other profiles, the available tasks are automatically created and configured.

Therefore, to add a new task (if the selected profile allows it), right-click on the Task Configuration and select the *Add Object*, then *Task* options, as shown on figure below:

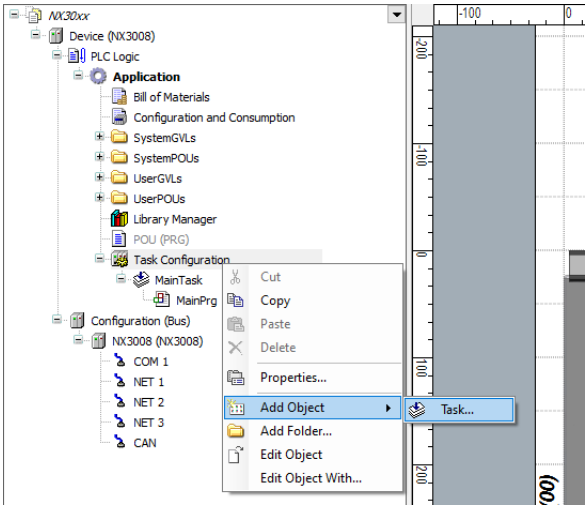

Figure 20: Creating a Task

Following, a screen will appear requesting the task name. After, click on *Add* to end the task creation.

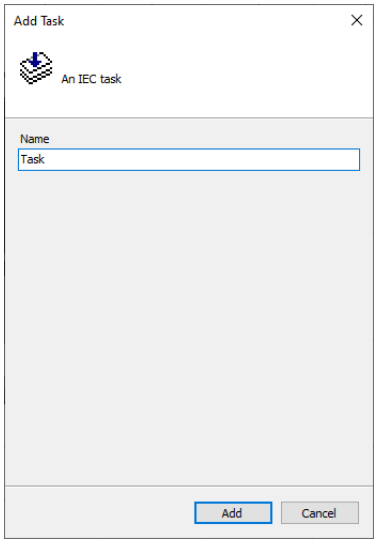

Figure 21: Task Name

4.4.1. Task Configuration

After the task is open, the configuration window will appear for the user to define and classify its functioning.

Priority (0...31) field defines the priority, a number between 0 and 31, where 0 is the highest priority and 31 is the lowest. For instance, the MainTask, created in the majority of the project profiles, has priority 13, so this task is considered to have high priority for the system.

Type field defines the type of the task. The selection list offers the following task types:

- *Cyclic*: The task is executed in cycles, or it is called every time interval configured in the field at its side. E.g.: t#20ms.
- *Event*: The task is executed when the variable is of the BOOL type, configured in the field at its side, receives a rise-going edge, in other words, the variable value goes from FALSE to TRUE.
- *External*: The task is executed when the external interruption occurs. It is configured in the field at its side.
- *Freewheeling*: The task is always executed, according to its priority, in other words, tasks with higher priority are executed first.
- *Status*: The task is executed when the BOOL variable, configured in the field at its side, is true.

In addition to the fields mentioned above, it is necessary to configure the *Interval* (mandatory for the cyclic type): the period of time after which the task will be called to run. The maximum time for the *MainTask* in the *Single*, *Basic*, *Normal*, *Expert* and *Custom Profiles* is 750 ms. In the *Machine profile*, the maximum time is 100 ms. The minimum time is 1 ms for all CPUs and profiles. It's recommended that the task interval be set to at least twice its cycle (execution) time.

The CPU watchdog is configured to prevent user tasks from stopping. The Time field sets the maximum time allowed for task execution. If the task takes more time than the watchdog to execute, the application will STOP and take exception by watchdog.

The Sensitivity field indicates how many times the watchdog must be reached to acknowledge the exception. When the execution time of the task reaches the value of the Sensitivity field multiplied by the Time field, the diagnosis is also displayed. It should be noted that the CPU watchdog is not used to protect the user's application from run-time peaks, but from crashes. Therefore, its time should be configured with a high value compared to the execution time of the task to which it is related. The ideal is to keep the average execution time of tasks in, at most, 50% of the time of the watchdog. This will reduce the chances of watchdog errors for any peak time of task execution.

In order to protect the system from possible configuration errors, MasterTool IEC XE checks all cyclic tasks during compilation, the watchdog (Software Watchdog) and the minimum and maximum limits of the task cycle time (Interval). It is important to emphasize that the user must be careful when changing the predefined values in the project profiles, as this may put the system execution at risk. Therefore, it is recommended to use the default values.

For further information see Task Editor.

The table below shows the verifications performed by MasterTool IEC XE for the *Interval* field configuration of the cyclic tasks, when the *Sensitivity* field is 1. For the *Custom Profile*, the consistency on the task interval and on the watchdog, time isn't performed.

Task	Type	Minimum Interval	Maximum Interval
MainTask	Cyclic	1 ms	750 ms
CyclicTask	Cyclic	5 ms	2147483 ms
TimeInterruptTask00	Cyclic	500 us	2147483 ms

Table 6: Maximum Allowed Configurations

4.4.2. POU – Task Connection

As described previously, for a POU to be executed in the application, it must be connected to a task. In the project profiles (without considering the Custom), the POUs are already associated to its respective tasks. However, in case the Custom profile is being used or new POUs are created, they must be connected to the tasks.

To associate a created POU, the desired task must be accessed through a double-click on it in the device tree, and the *Add Call* option selected. After this, a screen called *Input Assistant* will appear on the screen where the desired POU must be selected, as shown on figure below:

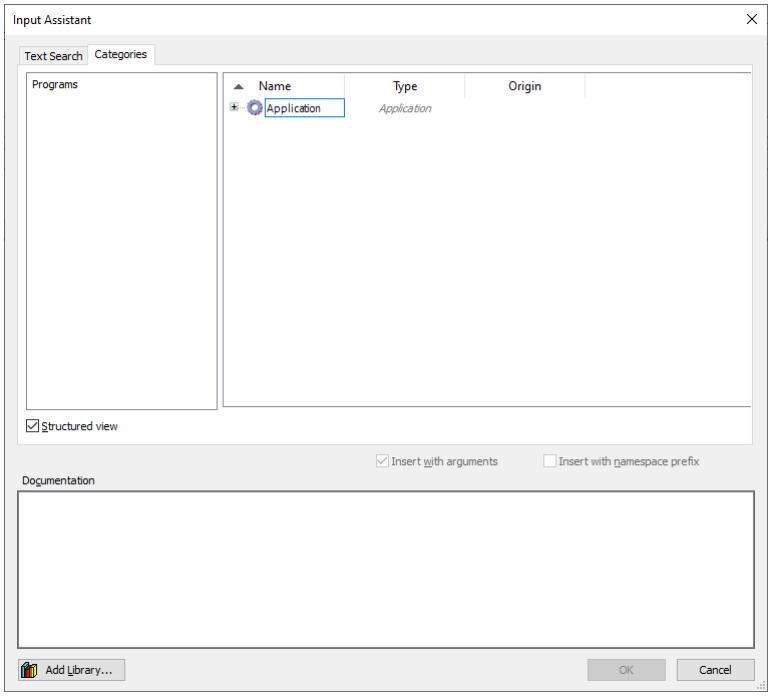

Figure 22: Connecting POUs to Tasks

4.4.3. Maximum Number of Tasks

The maximum number of tasks the user can create is only defined for the Custom profile, the only one that has this permission. The others have already created and configured their tasks.

For more information about the maximum number of IEC tasks per CPU and project profile, refer to the specific manual for each CPU.

4.5. CPU Configuration

The Nexto CPU configuration is based on the action of structuring the diagnostics area, the retentive and persistent memory area and hot swap mode, among other parameters.

The user must double-click on the CPU icon (located in the device tree) and configure the field as described in the Editors chapter of this manual and on the manual of the CPU.

4.6. Libraries

There are several programming tool resources, which are available through libraries. Therefore, these libraries must be inserted in the project so its utilization becomes possible. The insertion procedure is rather simple: the user must select the item *Library Manager*, available in the left menu and select *Add library*, as shown in the figure below:

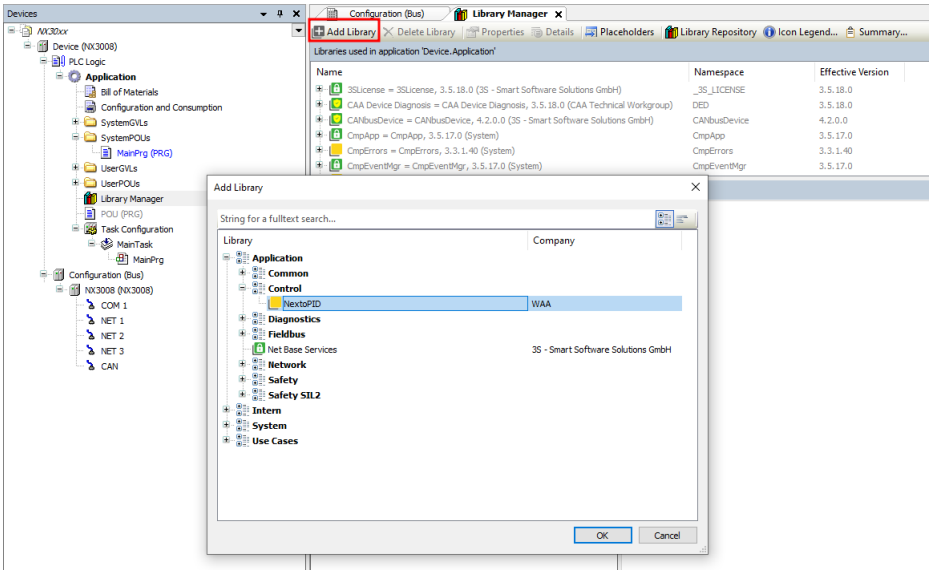

Figure 23: Inclusion of a Library in the Project

After, the desired library must be selected for project inclusion, when the button *OK* must be pressed.

Note: Depending on the selected options, this dialog may have more or less options.

4.7. Inserting a Protocol Instance

The Nexto Series CPUs offer protocols as the MODBUS. The desired protocol instance must be added and configured in the communication interface.

Two cases of MODBUS protocol insertion are described below: one in the serial interface and the other in the Ethernet interface.

4.7.1. MODBUS RTU

The first step for the MODBUS RTU configuring, in slave mode, is to include the instance in the desired COM (COM 1 in this case) by clicking with the right button on the COM and select *Add Device...*, as shown in the figure below:

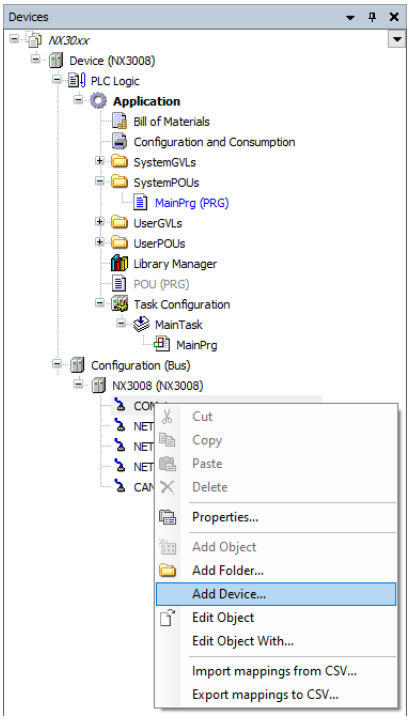

Figure 24: Adding an Instance

After that, the available protocols for the user will appear on the screen. Set the mode of the protocol configuration and select *MODBUS Symbol RTU Slave* or *MODBUS RTU Slave* for the correspondent configuration. Then, click *Add Device*, as shown in the figure below:

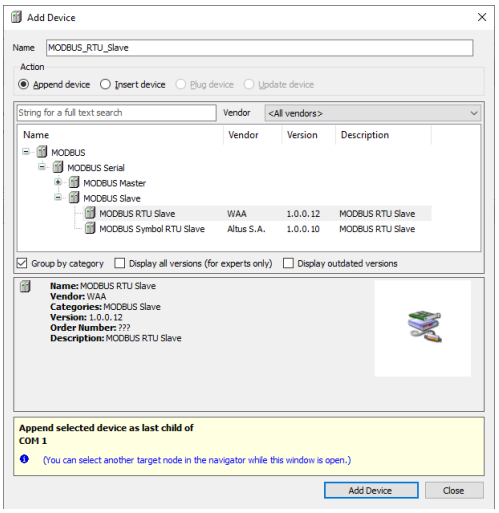

Figure 25: Selecting the Protocol

4.7.2. MODBUS Ethernet

The first step to configure the MODBUS Ethernet, in client mode, is to include the instance in the desired NET (in this case, NET 1. Click on the NET with the mouse right button and select *Add Device...*, as shown in the figure below:

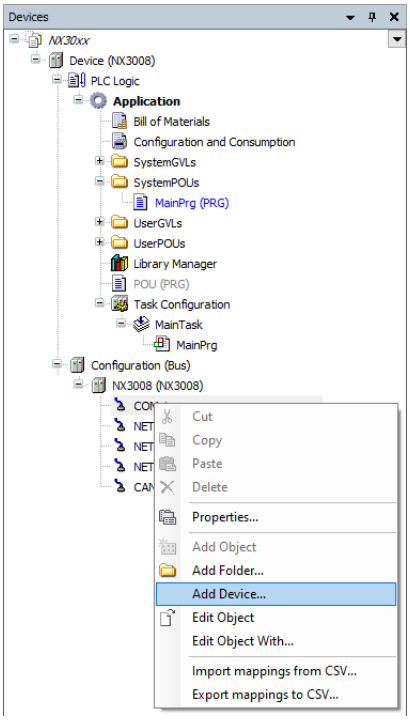

Figure 26: Adding an Instance

After that, the available protocols for the user will appear on the screen. Set the mode of the protocol configuration and select *MODBUS Symbol Client* or *MODBUS Client* for the correspondent configuration. Then, click *Add Device*, as shown in the figure below:

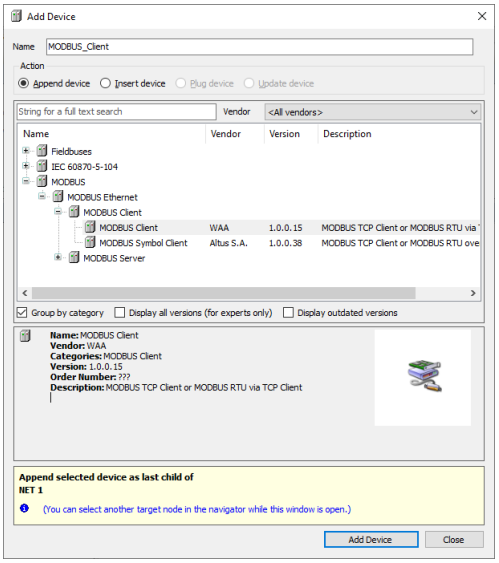

Figure 27: Selecting the Protocol

4.8. Building a Project

In order to execute the verification of the created application, the user must compile the project. This is the most efficient way for finding or receiving error warnings regarding any mistake made during the product configuration and application edition. To execute such procedure, access the *Build* menu and click on option *Generate code*.

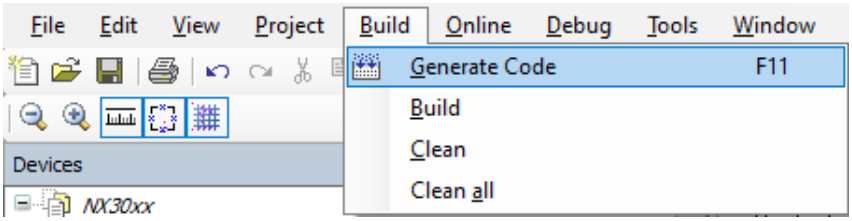

Figure 28: Building the Project

After the processing time, which varies according the user application size, the errors and alarm messages, in case they are needed, will be shown below, as depicted in the figure below:

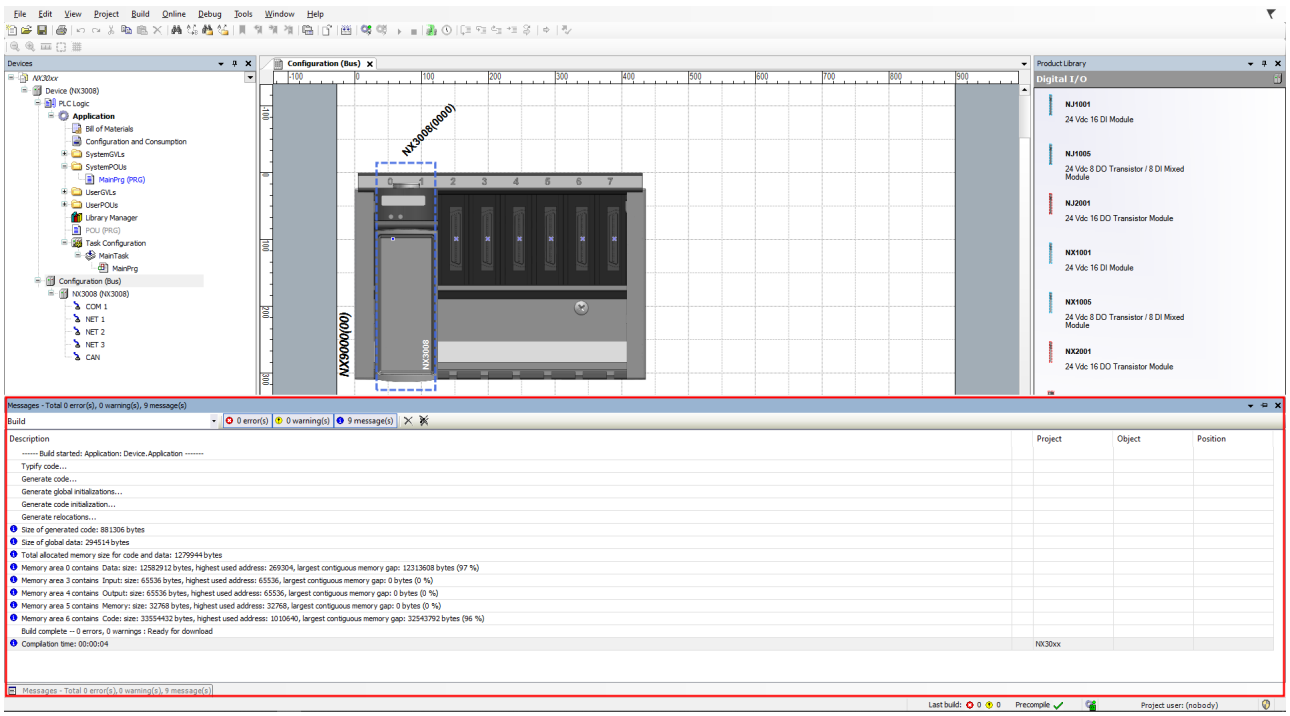

Figure 29: Building Messages

In case the errors and messages are not visible on the screen, the option *Messages* from the menu *View* must be selected, as shown in the figure below:

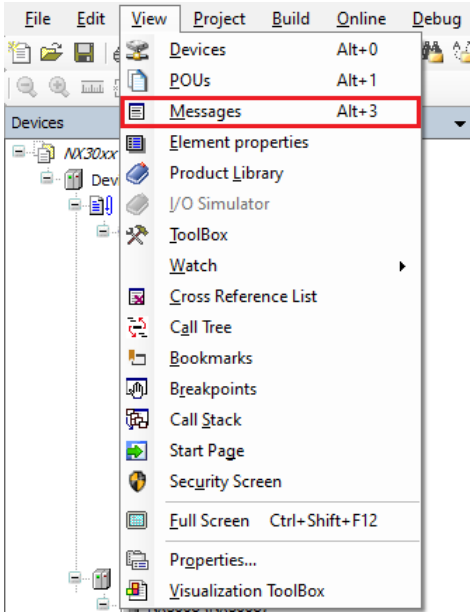

Figure 30: Including the Messages on the Screen

4.9. Simulation Mode

MasterTool IEC XE has an important simulation feature which allows the user to test his application without the equipment use, turning possible a higher flexibility at the program development. However, some specific resources, depending on the Nexto CPUs hardware, are not possible to be simulated.

The resources unavailable in the simulation mode are the following:

- RTC Watch
- Bus scan
- I/O Modules
- Bus interrupt
- Serial Ports
- Ethernet Communication
- Communication protocol as Modbus
- PROFIBUS Interfaces
- PROFIBUS Slaves
- SD card operation
- Variables diagnosis
- Communication protocol as Modbus
- Other functions that access the hardware of PLC

For this reason the simulation mode should be used to test the application logic in that do not depend on hardware access functions. These resources should be tested with the hardware to ensure the functioning of the application in this sense.

To change the MasterTool IEC XE to simulation mode you must select this option in the *Online* menu as in the figure below. After that a warning is displayed in the bottom bar of MasterTool IEC XE that indicates that the tool is operating in Simulation Mode.

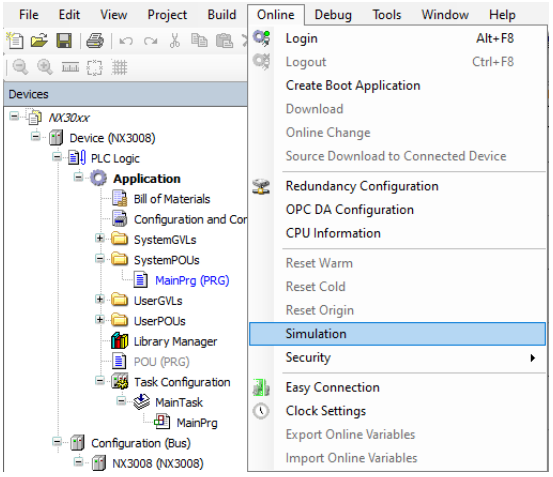

Figure 31: Simulation Mode

In Simulation Mode the application runs in a virtual device on the computer where it is installed the MasterTool IEC XE . For this reason some features displayed are related to computer hardware and not Nexto Series CPUs architecture. The main feature in this sense is related to the format of the data in the memory areas of direct representation. The Simulation Mode works with the little endian format where the first memory address is the least significant of data. On the other hand, some Nexto Series CPUs work with the big endian format where the first memory address is the most significant data.

In this case the same data as written for example in %QD0, will be written differently in simulation and in Nexto series CPUs. If the data written is 16#1234ABDC the distribution of the data in memory of PLC shall be as follows:
%QW0 = 16#1234
%QW2 = 16#ABCD
%QB0 = 16#12
%QB1 = 16#34
%QB2 = 16#AB
%QB3 = 16#CD

For the same data written in %QD0 in simulation mode the distribution of data in memory will be as follows:
%QW0 = 16#ABCD
%QW2 = 16#1234
%QB0 = 16#CD
%QB1 = 16#AB
%QB2 = 16#34
%QB3 = 16#12

In view of these differences and to facilitate application development, using the capabilities of MasterTool IEC XE and Nexto Series CPUs is recommended the use of symbolic variables. In this case the differences between the simulation and the behavior with the Nexto Series CPUs are not checked. Therefore, the best practice is to avoid the use of direct representation variables whenever possible to avoid rework when developing a logic that will be tested in simulation and then loaded into a CPU.

Simulation can be used to simulate a redundant project, however, it's going to have the same limitations as mentioned earlier, only enabling the test of logic that doesn't depend on the hardware. In this case, the POUs NonSkippedPrg and ActivePrg will always be executed, as if the simulated PLC is the Active one.

It is also important to stress due to differences between the architectures of the devices that the same code generated by using the simulation device can have sizes - in the areas of data and code - other than those generated for a Nexto series CPU.

4.10. Create and Run Projects

See in the following a description of how to create a simple project containing a PLC program, further how to load this program via a Gateway Server to the PLC (target device) and to get it run and monitored. The PLC runtime system used for this example project by default is provided with the MasterTool IEC XE setup.

The sample program will be written in Structured Text language (ST) and consist of a program (UserPrg) and a function block (FB1). UserPrg will contain a counter variable (ivar) and call function block (FB1). FB1 will get input *in* from UserPrg, will add *2* on this input and will write the result to an output *out*. UserPrg will read *out*.

Notice that the following descriptions refer on the default configuration of the user interface provided with the currently installed version of the programming system.

- Start MasterTool IEC XE

- Create a project
- Declare variables in UserPrg
- Enter Programming Code in the Body of UserPrg
- Create a Programming POU (ST function block FB1)
- Define the Resources for Running and Controlling the Program
- Configure a Communication Channel to the CPU
- Compile and Load Application on the CPU
- Starting the Application
- Debug an application
- Breakpoint settings and program scan

4.10.1. Declare Variables in UserPrg

The POU *UserPrg*, which by default is already available in the Devices window, is automatically created in any language the user select. It can be found in the folder *UserPOUs*, and with a double-click the language editor window will open in the center part of the MasterTool IEC XE user interface. In this case, will be used the ST language.

Basically, a POU always can be opened in its editor view by a double-click on the entry in the devices POUs tree.

The ST editor consists of a declaration part (upper) and a *body* (lower part), separated by a movable screen divider.

The declaration part shows line numbers at the left border, the POUs type and name (PROGRAM UserPrg) and the embracing keywords *VAR* and *END_VAR* for the variables declaration.

The body is empty, only line number 1 is displayed in the figure below:

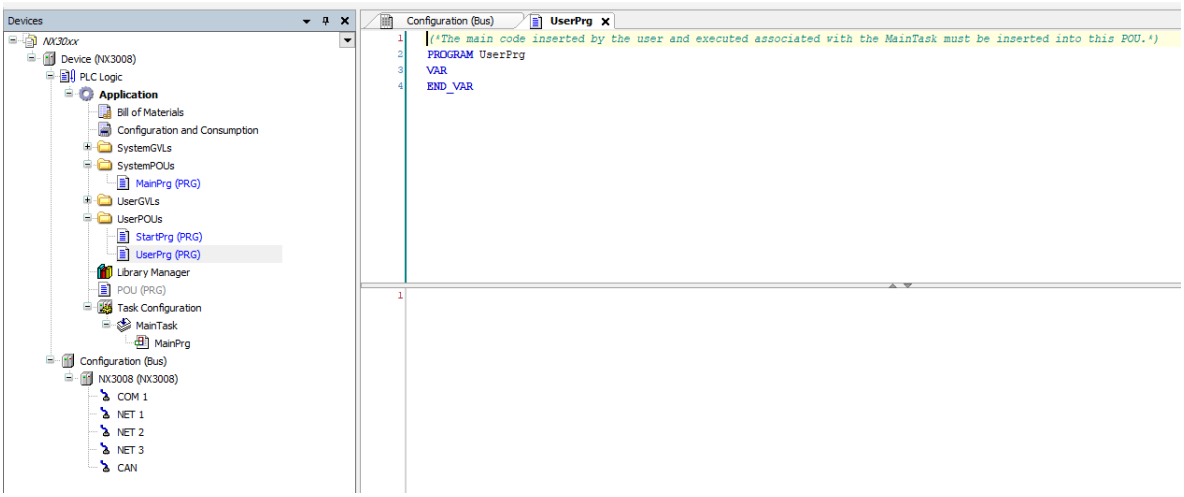

Figure 32: ST Editor Window

In the declaration part of the editor put the cursor behind VAR and press the <ENTER> key. An empty line will be inserted where you may enter the declaration of the variables *ivar* and *erg* of type INT and fbinst of type FB1:

```
PROGRAM UserPrg
VAR
    ivar: INT;
    fbinst: FB1;
    erg: INT;
END_VAR
```

Alternatively, you may directly type an instruction in the implementation part of the editor (body) and make use of the Autodeclare function

4.10.2. Enter Programming Code in the Body of UserPrg

```
ivar := ivar+1; // counter
fbinst(in:=11, out=>erg); // call function block of type FB1
// with input parameter \textit{in}
// output is written to \textit{erg}
```

Instead of steps, you can use the Auto Declaration feature: Without a preceding declaration, type a statement directly into the body of the program, then press the <CTRL> + <.> (or just put the on it and click in the button that will appear) for each undeclared variable found in the implementation line. The Auto Declare dialog will open, where you can now make the declaration settings. This opening process can be viewed in the figure below:

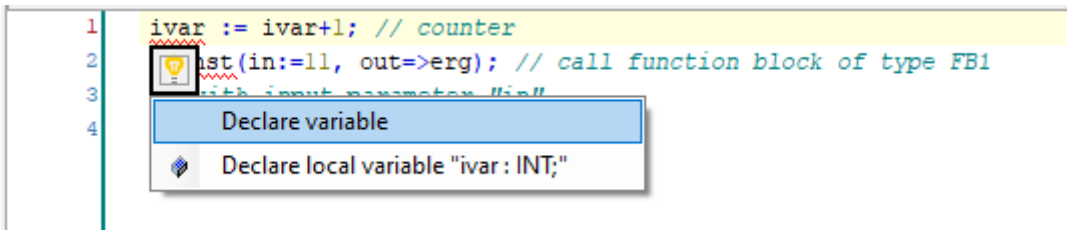

Figure 33: Auto Declare Opening

The Auto Declare window can be seen the in the figure below:

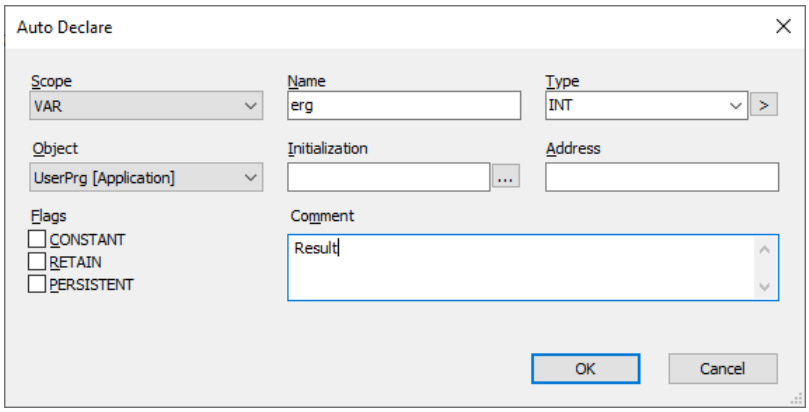

Figure 34: Auto Declare Dialog

The variables name and scope as well as the current POU (Object) will be filled in automatically. Enter the desired type and initialization value according to the declaration described above and add a comment. Additionally you could insert a direct representation variable address in the Address field.

Confirm the dialog with *OK*. This will enter the declaration of *erg* in the declaration part of the POU with the comments, as shown in the figure below:

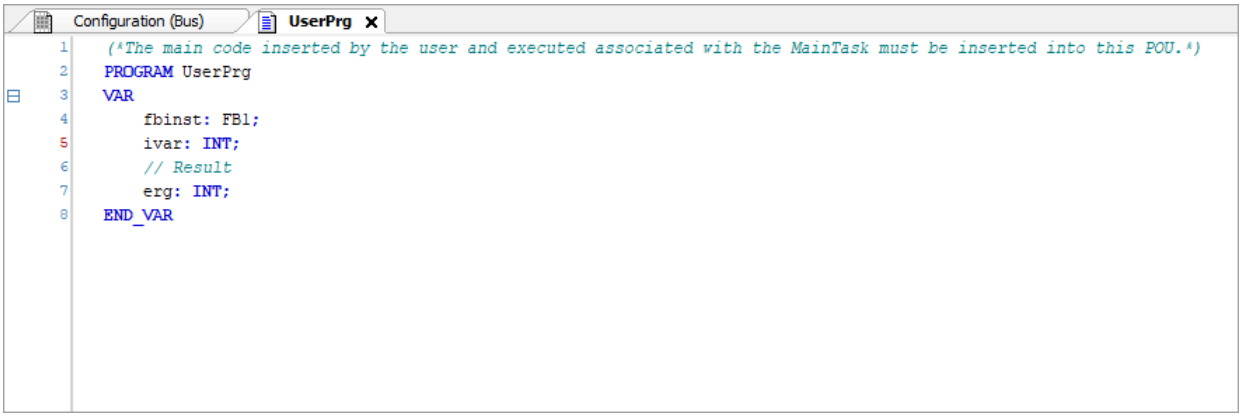

Figure 35: erg Variable Declaration

4.10.3. Create a Programming POU (ST function block FB1)

We supply another function block FB1, which will add *2* on the input given by variable *in*: out = in + 2.

Choose command *Add object* from the *Project* menu.

Select POU in the left part of the *Add Object* dialog. Enter the name *FB1* for the POU and activate option Function Block in the Type section.

Choose Structured Text (ST) for the Implementation language.

Press button *Add* to confirm the object settings.

A further editor window will open for the new function block FB1. Declare there in the same way as done for UserPrg the following variables:

```
FUNCTION_BLOCK FB1
VAR_INPUT
  in:INT;
END_VAR
VAR_OUTPUT
  out:INT;
END_VAR
VAR
  increment:INT:=2;
END_VAR
```

In the implementation part of the editor enter the following:

```
out:= in+increment;
```

4.10.4. Define the Resources for Running and Controlling the Program

4.10.4.1. Start Gateway Server

The Gateway Server is started automatically at system start as a service. Make sure that there is an icon ( ) in the system tray, indicating that the gateway is running. If the icon is looking like ( ), the gateway is currently stopped. This icon is part of the GatewaySysTray program, which is available for controlling and monitoring the Gateway service. It provides a menu with a *Start Gateway* and *Stop Gateway* commands, thus allowing the user to stop or restart the service manually. The menu also includes the command *Exit Gateway Control*, which just terminates the GatewaySysTray program, not however the Gateway service. The GatewaySysTray program is started automatically when Windows is started, however it also can be started manually via the *Programs* menu.

4.10.4.2. Configure a Communication Channel

The resulting connection is then entered in the line below Select network path to the *CPU controller*.

In the *Devices window*, double-click on the *Device* entry. The Device dialog opens with the *Communication Settings* subdialog. Here you have to set up the connection between the CPU and the programming system according to the item Find the network.

4.10.5. Run and Watch the Application

4.10.5.1. Compile and Load Application

If you just want to check your active application program for syntactic errors, perform command *Generate Code* (*Build* menu).

Note: No code will be generated in this case. Information, warnings and error messages will be displayed in the Messages window which is placed at the lower part of the user interface by default).

Even if this syntactical check has not been done before, you can log into the CPU. Therefore make sure, that CPU is running (that is the symbol in the system bar is colored).

Use command Login (*Online* menu). If the communication settings have been properly configured the following message box will appear (otherwise, the user will be asked to correct the communication settings):

There is no device application on target. Do you want to create it and proceed with download?

Confirm with *Yes* to start the compilation and download of the application.

The compile messages will be displayed in the *Messages* window. If the project has been created correctly, no compilation errors are to be expected, so that the application can now be started.

4.10.5.2. Starting the Application

Perform command *Start* (*Debug* menu). In consequence the program starts running. A green RUN will be displayed in the status bar at the bottom of the user interface.

4.10.5.3. Monitoring the Application

There are three possibilities for watching the variables of the application program:

- Watch lists
- Writing and force values
- Online views of the POUs Watch lists

4.10.5.3.1. Open an Instance Window of the Program

The instance view of a POU provides all watch expressions of that instance in a table view in the declaration part and – if activated – as *inline monitoring* also in the implementation part.

In order to open the online view, perform a double-click on POU *UserPrg* in the *Devices* window or select this entry and choose command *Edit Object* (context menu).

In the lower part of the view, the user can see the code lines as entered in offline mode, supplemented by the little inline monitoring windows behind each variable, showing the actual value. In the upper part, a table shows the watch expressions of the POU, that is the current values of the application variables on the CPU.

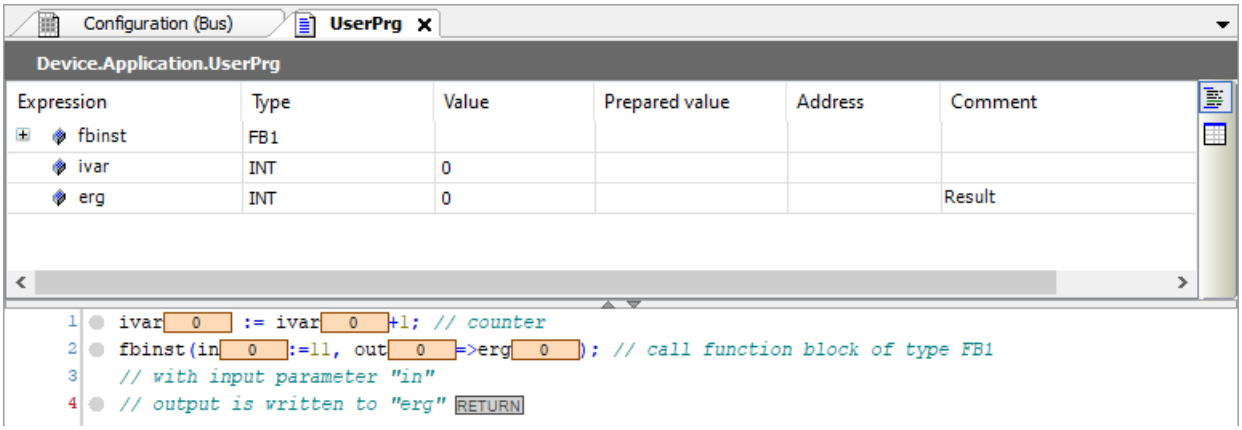

Figure 36: Monitoring Expressions and Code Lines

4.10.5.3.2. Writing and Forcing Variables

You can write or force a *Prepared value* to the *ivar* variable on the CPU, which means that *ivar* will be set to that value at the start of the next cycle. Double-click on the field in the *Prepared value* column, enter the desired integer value and exit the field by pressing <ENTER> or by clicking outside the field. Use the *Write Values* and *Force Values* command (*Debug* menu) to write or force this value to the CPU. You will immediately see the result in the Value column.

4.10.5.4. Using the Watch Views

Watch view windows can be used to configure specific sets of watch expressions of the application, for example for debugging purposes.

From the *View* menu use the commands *Watch*, then *Watch 1*. The Watch window opens.

In the *Expression* column, click on the first line of the table to open an edit box. Enter the full path for the variable *ivar* to be monitored: *Device.Application.UserPrg.ivar*.

It is recommended to use the input wizard via button for this purpose. Close the edit box with <ENTER>. The type is added automatically.

Do the same for the other variables. The watch list shown in the next picture contains only expressions of UserPrg, but of course the user can create a set of any variables of your project. Note that for instance variables, such as the FB1 instances, it is sufficient to enter the expression *Device.Application.UserPrg.fbinst*. The respective variables are automatically entered and the corresponding lines can be opened with the + symbol: The current value of a variable is displayed in the *Value* column:

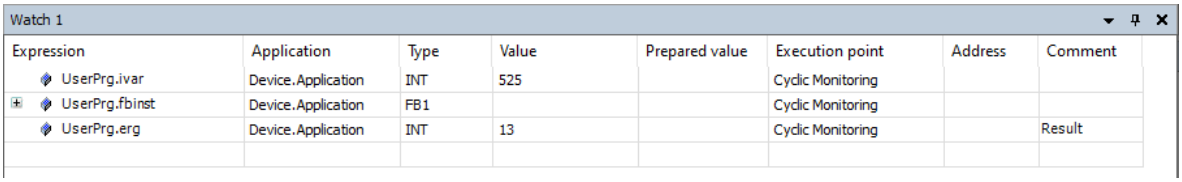

Figure 37: Watch List

If not yet done, now select the application object and perform command *Start* from the context menu. The application will be started on the CPU and the current value will be displayed in column *Value*:

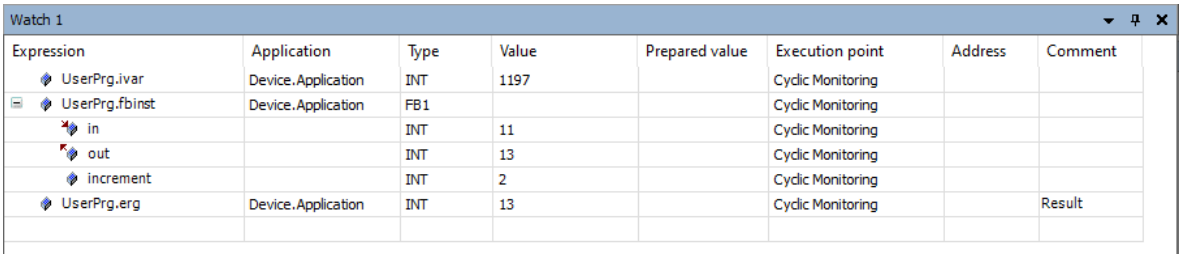

Figure 38: Current Value

Writing and forcing values is also possible.

To disconnect from CPU perform command *Logout* from the *Online* menu.

4.10.6. Debug an Application

4.10.6.1. Set Breakpoint and Step Through the Program

In online mode, you can set breakpoints as defined stops for program execution.

When the program reaches a breakpoint, you can execute the program in steps. At each stop position, you will see the current value of the variables in the monitor views.

Select line 1 of UserPrg. Press <F9>, which is equivalent to the *Toggle Breakpoint* command on the *Debug* menu. The breakpoint is displayed.

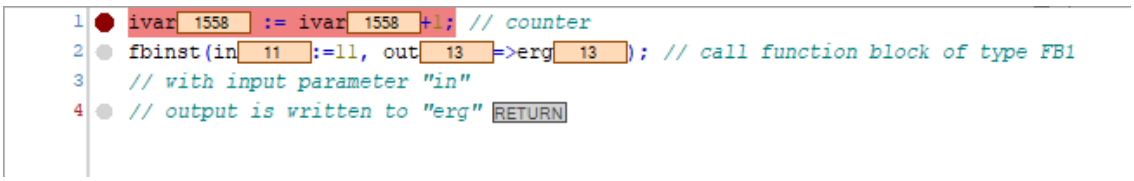

Figure 39: Application in Stop State

A running application will stop at a breakpoint:

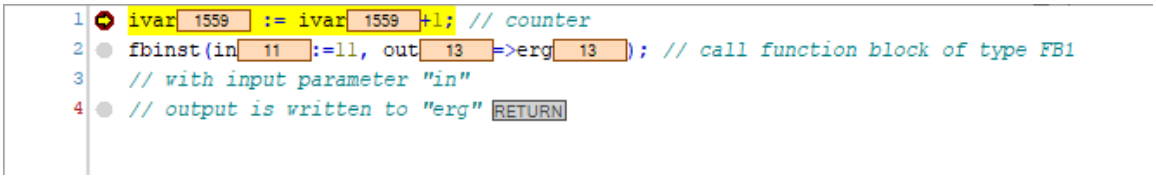

Figure 40: Application in Run State

Now you can step further by using <F8>, which is the *Step Into* command from the *Debug* menu, which will also step into the function block instance.

To skip the steps of the function block, use <F10> or the *Step Over* command. Each variable value currently read from the CPU is displayed.

You may also want to take a look at the breakpoints dialog, which can be opened with the *Breakpoints* command from the *View* menu. Here you can view and edit the currently set breakpoints, and you can enter new breakpoints.

Also note that the breakpoint positions are remembered when you log out. They are indicated by faded red bullets.

4.11. Help

Currently a non-dynamic version of online help is installed. By default, it can be accessed via the *Help* menu. The language in which the help pages are displayed can be changed in the *Options* menu, *International Settings* dialog.

4.11.0.1. Context Sensitive Help

Default Shortcut: <F1>

You may press <F1> in within an active window, a dialogue or on a menu command to open the online help.

If a menu command is selected, the corresponding help page will be displayed.

Likewise, <F1> is executed on a selected text (for example a key word, a basic function or an error message within the message window), the corresponding help page will be displayed.

4.12. Uninstallation, Update, Repair

To uninstall the programming system and its additional components or to modify the current installation executes the current setup file.

5. User Interface

The following chapter describes the MasterTool IEC XE programming system user interface.

- User Interface Components
- Customizing the User Interface
- View objects in online mode

5.1. User Interface Components

The MasterTool IEC XE programming interface is an arrangement of *components*.

This interface depends on the arrangement of windows, which can be modified by the user at any time by moving, docking/undocking views, resizing or closing windows.

The user interface provides menus and toolbars, editor and object organization, watch and message windows, and an information and status line. This interface can be better seen in the figure below:

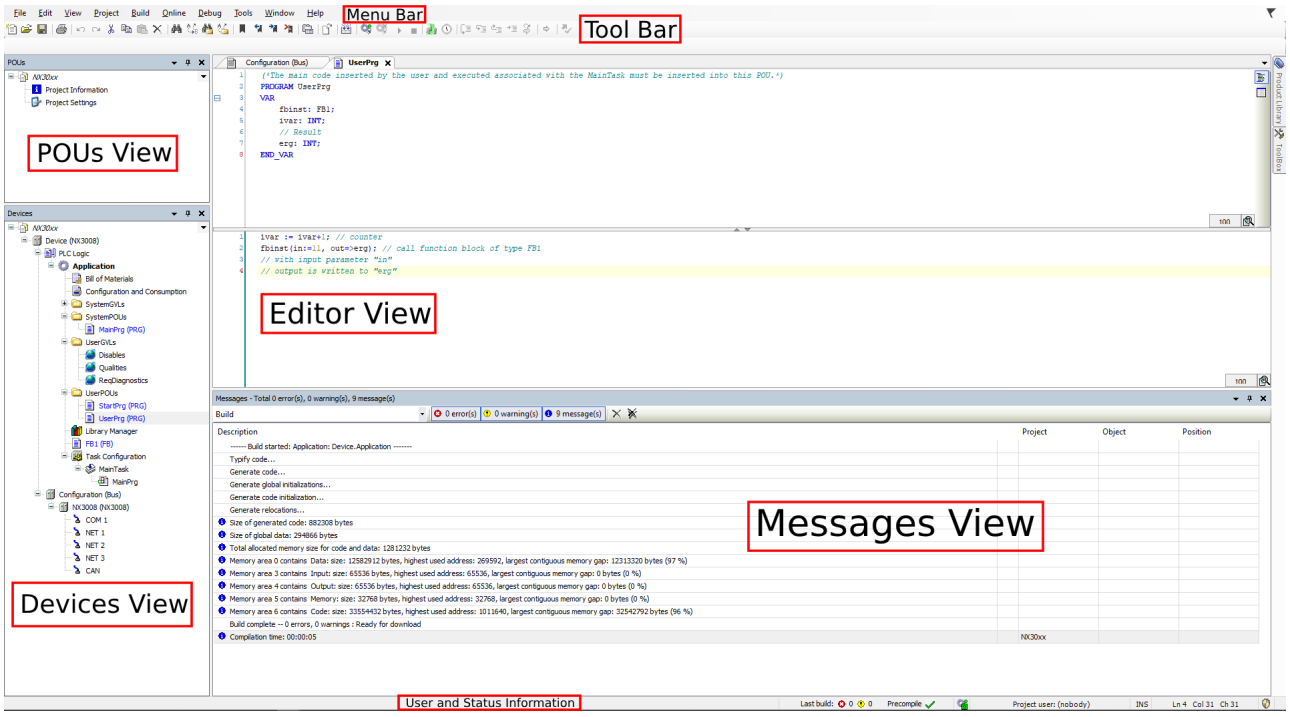

Figure 41: Example of the Product User Interface

The standard components:

- *Menu Bar*: Provides menus, which contain all currently available commands.
- *Tool Bar*: Contains tool buttons for all currently available tools provided by a toolbox plug-in.
- *POUs view*: For organizing the programming units (POUs, DUTs etc.) of a project in a tree structure (open from *View* menu).
- *Devices view*: For organizing the device resource objects of a project in a tree structure (open from *View* menu).
- *Editor view*: Used for creating the particular object in the respective editor. In case of language editors (for example ST-Editor, CFC-Editor) usually the window combines the language editor in the lower part and the declaration editor in the upper part. In case of other editors, it also can provide dialogs (for example Task-Editor, Device-Editor, CPU Editor). The POU's or Resource object's name always is displayed in the title bar of the window. The objects can be opened in the editor window in offline or online mode via command Edit Object.

For information on what is currently going on in your project in off-line or online mode see the following components:

- *Messages view*: Precompile, Compile, Build, Download messages etc. are displayed in this window.

- *Watch view and Online Views of Editors*: Shows a monitoring view of a POU and a user-defined list of watch expressions.
- *Information and Status Line*: The line at the lower border of the user interface provides information on the currently logged-in user. Also - if the user are currently working in an editor window - the current position of the cursor and the status of editing mode. In online mode the current status of the program will be indicated.
- *Project user*: Each project has a user and access management. The currently logged-in user will be named in the status line.
- *Position*: Counted from the left and upper margin of the editor window:
  - Ln = Number of lines.
  - Col = Number of columns (a column includes exactly one space, character or digit).
  - Ch = Number of characters (in this context a character can be a single character or digit as well as a tab including for example 4 columns).

By a double mouse-click on one of the fields, the user get the dialog *Go To Line*, where you can enter a different position where the cursor should be placed.

- *Status of editing mode*: INS refers to insert mode and OVR refers to overwrite mode. By a double mouse-click on this field you can toggle the setting.
- *Online mode information*: status of the application on the device:
  - RUN = program running.
  - STOP = program stopped.
  - HALT ON BP = program halted on a breakpoint.
  - Program loaded = program loaded on device.
  - Program unchanged = program on device matches that in the programming system.
  - Program modified (Online Change) = program on device differs from that in the programming system, online change required.
  - Program modified (Full download) = program on device differs from that in the programming system, full download required.

5.1.1. Windows, Views, Editor Windows

The windows that are displayed within or next to the user interface frame window all look the same at first glance. However, there are two types:

- Some can be docked to any edge of the frame window, or alternatively can be positioned on the screen as undocked windows independent of the frame window. In addition, they can be *hidden*, i.e. represented only by a tab in the frame border. These windows display information that does not depend on a single object of the project, e.g. Messages, Devices, POUs, Toolbox. They can be accessed using the *View* menu commands and are also called *Views*. Most views include a non-configurable toolbar with buttons for sorting, viewing, and searching within the window.
- Others open when you view or edit a specific project object in the corresponding editor. These appear in a tabbed editor pane or as MDI windows, depending on your interface settings. They cannot be *hidden* or undocked from the frame window. They can be accessed using the *Window* menu commands.

Additional window types or views can be added using vendor-specific components.

5.2. Customizing User Interface

The actual appearance of the user interface, that is, the arrangement and configuration of each component, depends on the following:

- Standard pre-settings for menus, keyboard functions and toolbars. MasterTool IEC XE install default settings.
- Properties of an editor as defined in the respective Options dialogs. In addition, these settings can be overwritten by the user and the current configuration also will be saved on the local system.
- Arrangement of views or editor windows within the project, done by the user. The current positions are saved with the project.

5.2.1. Zoom

Each editor window has a zoom function. The zoom button ( ) in the lower right corner of the window opens a list from which you can select one of the following zoom levels: 25, 50, 75, 100, 150, 200, and 400 percent. Note that a printout always refers to the 100% view. The user can also customize the zoom level by entering the desired value in the appropriate field.

5.3. User Interface in Online Mode

As soon as you log in with the project, all objects that have already been opened in offline mode are automatically displayed in online mode.

To open an object in online mode that has not already been opened in offline mode, double-click the object entry in the POUs or Devices window or use the *Edit Object* command.

If your selection is unique, the object is opened in online mode. Otherwise, for example, if there are multiple instances of the selected object (function blocks, etc.) in the project, a *Select Online State <object name>* dialog appears, allowing you to choose whether to view an instance or the base implementation of the object, and whether to view the object in online or offline mode.

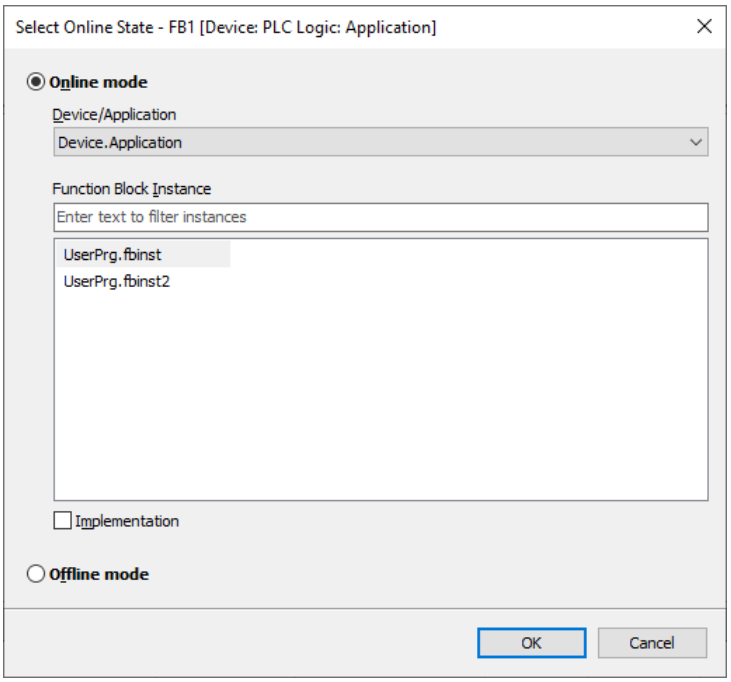

Figure 42: Select Online State Dialog

In the *Device/Application* field see the Device and Application to which the respective object is associated.

To open the online view of the object activate option *Online* mode and press *OK*. To see the offline view activate option *Offline* mode.

If the object is a function block in the *Function block instance* field there will be a list of all instances used in the current application. In this case, the user can:

- Select one of the instances and enable online or offline modes;
- Select the Implementation option, which, regardless of the selected instance, open the basic view of the implementation of the function block. The implementation has no effect for objects not instantiated.

For further information on the online views of the particular editors refer to the respective editor descriptions.

The status bar will provide information on the current status of the application.

5.4. Standard Menus and Commands

The following is an overview of the structure of the main menus and commands.

The special commands for a certain editor are usually available in a corresponding menu, which is available when the editor is opened (e.g., when you edit an object in the SFC editor, the SFC menu is added to the menu bar).

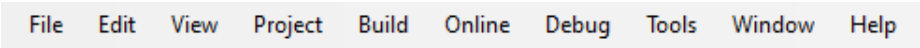

Figure 43: Standard Menu Bar

5.4.1. Standard Menus and Commands

5.4.1.1. File Menu

Commands to perform actions on the project file (open, close, save, print, page setup, source download/upload...).

Symbol	Command	Shortcut
	New project...	<Ctrl>+<N>
	Open Project...	<Ctrl>+<O>
	Close Project	
	Save Project	<Ctrl+<S>
	Save Project as...	
	Project Archive	
	Extract Archive...	
	Save/Send Archive...	
	Source Upload...	
	Source Download...	
	Print...	
	Print Preview...	
	Page Setup...	
	Recent Projects	
	<n> <Project Path>	
	Exit	<Alt>+<F4>

Table 7: File Menu

5.4.1.2. Edit Menu

Commands available for working in editors (language editors, declaration editor).

Symbol	Command	Shortcut
	Undo	<Ctrl>+<Z>
	Redo	<Ctrl>+<Y>
	Cut	<Ctrl>+<X>
	Copy	<Ctrl>+<Ins>
	Paste	<Ctrl>+<V>
	Delete	<Del>
	Select All	<Ctrl>+<A>
	Find Replace	

Symbol	Command	Shortcut
	Find	<Ctrl>+<F>
	Replace	<Ctrl>+<H>
	Find in Project	<Ctrl>+<Shift>+<F>
	Replace in Project	<Ctrl>+<Shift>+<H>
	Find Next	<F3>
	Find Next (Selected)	<Ctrl>+<F3>
	Find Previous	<Shift>+<F3>
	Find Previous (Selected)	<Ctrl>+<Shift>+<F3>
	Toggle Field for Incremental Search	<Ctrl>+<Shift>+<I>
	Browse	
	Go to Definition	
	Display Cross References	
	Display Call Tree	
	Insert File As Text...	
	Advanced	
	Overwrite Mode	<Ins>
	View Whitespace	
	View Indentation Guides	
	Go to Line...	
	Make Uppercase	<Ctrl>+<Shift>+<U>
	Make Lowercase	<Ctrl>+<U>
	Go to Matching Bracket	
	Select to Matching Bracket	
	Expand All Folds	
	Collapse All Folds	
	Comment out selected lines	<Ctrl>+<O>
	Uncomment selected lines	<Ctrl>+<I>
	Bookmarks	
	Toogle Bookmark	<Ctrl>+<F12>
	Next Bookmark (Active Editor)	<F12>
	Previous Bookmark (Active Bookmark)	<Shift>+<F12>
	Clear All Bookmarks (Active Editor)	
	Input Assistant...	<F2>
	Auto Declare...	<Shift>+<F2>
	Next Message	<F4>
	Previous Message	<Shift>+<F4>
	Go To Source Position	
	Rename <variable>...	
	Add Variable...	

Symbol	Command	Shortcut
	Remove <variable>...	
	Reorder Variables...	
	Update Referenced Pins	

Table 8: Edit Menu

5.4.1.3. View Menu

Commands for activating the respective default views, i.e. for displaying them in a window in the user interface.

Symbol	Command	Shortcut
	POUs	<Alt>+<0>
	Devices	<Alt>+<1>
	Messages	<Alt>+<3>
	Element Properties (for SFC and Visualization elements)	
	Product Library	
	I/O Simulator	
	ToolBox	
	Watch	
	Watch <n>* (* *n* goes from *1* to *4*)	
	Watch all Forces	
	Cross Reference List	
	Call Tree	
	Bookmarks	
	Breakpoints	
	Call Stack	
	Start Page	
	Security Screen	
	Full Screen	<Ctrl>+<Shift>+<F12>
	Properties...	
	Visualization ToolBox	

Table 9: View Menu

5.4.1.4. Project Menu

Commands for handling project objects and the general information on the project.

Symbol	Command	Shortcut
	Add Object	
	Add Folder...	
	Scan for Devices...	
	Update Device...	
	Edit Object	
	Edit Object With...	
	Set Active Application	
	Project Information...	
	Project Settings...	
	Project Update...	
	Document...	
	Compare...	
	Commit Accepted Changes	
	Export PLCopenXML...	
	Import PLCopenXML...	
	Restore Points	
	Import Safety Project...	
	Delete Safety Objects Imported...	
	User Management	
	User Login...	
	User Logout	
	Permissions...	

Table 10: Project Menu

As can be seen in the table above, there's the command *Add Object*. With this command, the following objects can be added:

Symbol	Object
	Alarm Configuration...
	Cam Table...
	CNC program...
	CNC settings...
	Communication Manager...
	Datalogger...
	DUT...
	External File...
	Global Variable List...

Symbol	Object
	Global Variable List (tasklocal)...
	Image Pool...
	Interface...
	Library Manager...
	MQTT Client...
	Network Variable List (Receiver)...
	Network Variable List (Sender)...
	Persistent Variables...
	PID Control...
	POU...
	POU for Implicit Checks...
	Recipe Manager...
	Symbol Configuration...
	Text List...
	Trace...
	Trend Recording Manager...
	Unit Conversion...
	Visualization...
	Visualization Manager...
	Action
	Method
	Property
	Transition

Table 11: Objects that can be added

5.4.1.5. Build Menu

Commands to build the project, i.e. to perform a pre-compilation run including a syntax check. There are also commands for deleting the last compile information (*Clean* command), which is important for online changes and offline code generation.

Symbol	Command	Shortcut
	Generate Code	
	Build	
	Clean	
	Clean All	

Table 12: Build Menu

5.4.1.6. Online Menu

Commands for logging in and out to/from the controller, for loading the project on the controller and for reset.

Symbol	Command	Shortcut
	Login	<Alt>+<F8>
	Logout	<Ctrl>+<F8>
	Create Boot Application	
	Download	
	Online Change	
	Source Download to Connected Device	
	Redundancy Configuration	
	OPC DA Configuration	
	CPU Information	
	Reset Warm	
	Reset Cold	
	Reset Origin	
	Simulation	
	Security	
	Logoff Current Device User	
	Easy Connection	
	Clock Settings	
	Export Online Variables	
	Import Online Variables	

Table 13: Online Menu

5.4.1.7. Debug Menu

Commands for controlling the program run on the controller (Start, Stop) and for debugging actions (Breakpoints, Stepping, Writing, and Forcing).

Symbol	Command	Shortcut
	Start	<F5>
	Stop	<Shift>+<F8>
	New Breakpoint...	
	New Data Breakpoint...	
	Edit Breakpoint...	
	Toggle Breakpoint	<F9>
	Disable Breakpoint	
	Enable Breakpoint	
	Step Over	<F10>
	Step Into	<F8>

Symbol	Command	Shortcut
	Step Out	<Shift>+<F10>
	Run to Cursor	
	Set Next Statement	
	Show Next Statement	
	Write Values	<Ctrl+<F7>
	Force Values	<F7>
	Unforce Values	<Alt>+<F7>
	Display Mode	
	Binary	
	Decimal	
	Hexadecimal	
	Create PLC Crash Report	

Table 14: Debug Menu

5.4.1.8. Tools Menu

Commands for opening tools, which serve to prepare the environment for working on a project (installation of libraries and devices, options for editors, loading & saving etc.)

Symbol	Command	Shortcut
	Library Repository...	
	Device Repository...	
	License Repository...	
	OPC UA Information Model Repository...	
	License Manager...	
	Start PACTware	
	Options...	
	Miscellaneous	
	Generate PROFINET Device from GSDML	
	Scripting	
	Execute Script from File...	
	Enable Script Tracing	

Table 15: Tools Menu

5.4.1.9. Window Menu

Commands for handling the windows in the user interface (arrangement, opening, closing etc.).

Symbol	Command	Shortcut
	Next Editor	<Ctrl>+<F6>
	Previous Editor	<Ctrl>+<Shift>+<F6>
	Close All Editors	

Symbol	Command	Shortcut
	Reset Window Layout	
	New Horizontal Tab Group	
	New Vertical Tab Group	
	Float	
	Dock	
	Auto Hide	
	Next Pane	<F6>
	Previous Pane	<Shift>+<F6>
	<1> Configuration (Bus)	
Editor Icon	<n+1> <window title> (application path)	
	Windows...	

Table 16: Window Menu

5.4.1.10. Help Menu

Commands for getting online help and information on the programming system.

Symbol	Command	Shortcut
	Contents	<Ctrl>+<Shift>+<F1>
	Index	<Ctrl>+<Shift>+<F2>
	Search	
	Contact support	
	Update Software License...	
	Documentation	
	Datasheets and Specifications	
	Programming Manual	
	User Manual	
	Release Notes	
	Altus	
	About...	

Table 17: Help Menu

5.5. User Files Memory

Nexto Series CPUs have a memory area destined to the general data storage, in other words, the user can store several project files in the CPU memory. This memory area varies according to the CPU model used.

This functionality is accessed with a double click over the *Device* item and then selecting the *Files* tab, as showed in the figure below:

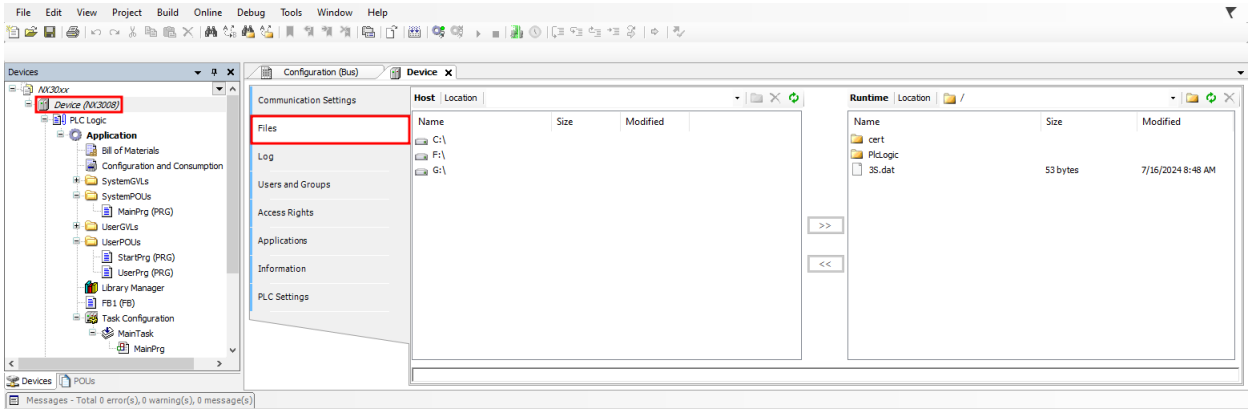

Figure 44: User Files Access

6. User and Access Right Management

6.1. User and Access Right Management of the Project

Provide functions to define user accounts and configure access rights within a project. Note that device-specific user management may also be supported to control the user's access rights to the PLC file system and objects at run time.

The rights to access project objects via specified actions are only assigned to user groups, not to an individual user account. Each user must therefore be a member of a group.

6.1.1. User Management

The configuration of users and groups is done in the *Project* dialog in the *Project Settings* window.

Automatically there are always the groups *Everyone* and *Owner*, that can't be deleted (just renamed).By default primarily each defined user of the group *Everyone*. Thus, each user account at least automatically is provided with defined default settings. No member can be removed from the group *Everyone*.

In the group *Owner* it automatically have the user *Owner*. Users can be added to or removed from this group, but at least one user must remain. This group also cannot be deleted and always has all access rights. Thus, it is not possible to make a project unusable by denying the respective rights to all groups. Both group and user *Owner* might be renamed.

When starting the programming system and a project, primarily no user is logged on the project. But then the user optionally might logon on via a defined user account with user name and password in order to have a special set of access rights.

Notice that each project has its own user management. Therefore, for example to get a special set of access rights for a library included in a project, the user must separately logon on to this library. In addition, users and groups, set up in different projects, are not identical even if they have identical names.

Note:

- The user passwords are stored irreversibly. If a password gets lost, the respective user account gets unusable. If the Owner password gets lost, the entire project might get unusable.
- By default, in new projects, the user *Owner's* password is empty.

6.1.1.1. Users

The currently registered users are listed in a tree structure. If defined via the *Add User* or *Edit User* dialog besides the Name (logon name) also the full name and a description for the user are displayed. The *ownerships* of each user can be *viewed/ hidden* via the plus/minus sign. Each user per default is member of group *Everyone*.

To define a new user account, use button *Add* to open the *Add User* dialog.

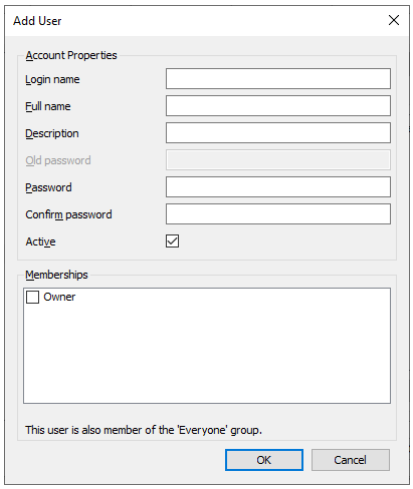

Figure 45: Add User Dialog

Fill in the following fields:

- *Logon name*: Logon name for the new user.

- *Full name*: Complete name of the new user. Just to give additional information.
- *Description*: Description on the new user. Just to give additional information.
- *Old password*: This field is only editable, when the dialog is used for modifying an existing user account. Before you can modify the password of an existing user, you must enter the currently valid password.
- *Password*: Password for the new user. The entry is masked by (*) characters.
- *Confirm password*: The entry made in field 'Password' must be repeated here and you will get an error message if the two entries do not match. In addition, here the entry is masked by (*) characters.
- *Active*: If this option is activated the user account is valid. If the account is not valid, the user cannot logon. An account might be deactivated automatically if repeatedly a logon has been tried with incorrect authentication entries.
- *Memberships*: In this list, all currently existing user groups are listed, beside group *Everyone*, to which the new user belongs automatically. By selecting the respective entries (☑) you can define to which groups the new user should belong.

To set up the new user close the dialog with *OK*. If there are incorrect entries (no login name, password mismatch, user already existing), you will get an appropriate error message.

To modify an existing user account: Use button Edit to open the *Edit User* dialog. The entry fields are the same as in the *Add User* dialog. The password fields however - for safety reasons - will show 32 * characters. After having modified the desired entries close the dialog with *OK* to get applied the new settings.

To remove one or several user accounts, select the respective users in the users list and use button *Remove*. Note that you will get no further inquiry. An appropriate error message will appear if you try to delete all users from the group. At least one must remain.

6.1.1.2. Groups

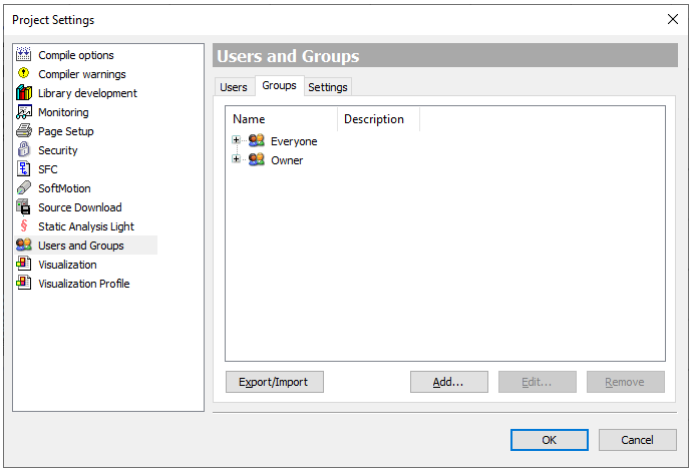

Figure 46: Project Settings Groups Dialog

The currently available groups are displayed in a tree structure. The members of each group can be *viewed/ hidden* via the plus/minus sign. A member again also might be a group. To add a new group: Use button *Add* to open the *Add Group* dialog.

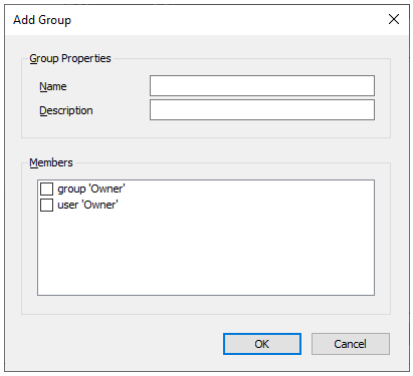

Figure 47: Add Group Dialog

Fill in the following fields:

- *Name*: Name for the new group.
- *Description*: Description on the new group. Just to give additional information.
- *Members*: List of all currently available users and groups. Select those ones (☑) which should be members of the current group.

To set up the new group close the dialog with *OK*. If there are incorrect entries (no name defined, group already existing, in Members having selected a group which would cause a *group cycle*, you will get an appropriate error message).

To modify an existing group: Use menu *Project*, click on the command *Project Settings* and select Users and Groups. So select the user you want to edit and click on the *Edit...* button. The entry fields are the same as in the Add Group dialog (figure above). The password fields however - for safety reasons - will show 32 * characters. After having modified the desired entries close the dialog with *OK* to get applied the new settings.

To remove one or several groups: Select the respective entries in the group's tree and use button *Remove*. Note that you will get no further inquiry! The members of the deleted groups will remain unmodified. An appropriate error message will appear if you try to delete the groups *Everyone* and/or *Owner*.

6.1.1.3. Settings

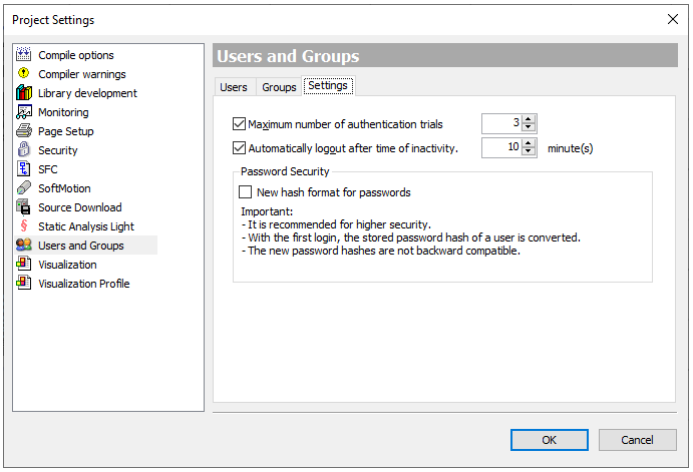

Figure 48: Add Group Dialog

The following basic options and settings concerning the user accounts can be made:

- *Maximum number of authentication trials*: If activated, the user account will be set invalid after the specified number of trials to log in with a wrong password. If not activated, the number of erroneous trials is unlimited. Default: option activated, number of trials: 3; permissible values: 1-10.
- *Automatically log out after time of inactivity*: If activated, the user account will be logged out automatically after the specified number of minutes of inactivity (no user actions via mouse or keyboard registered in the programming system). Default: option activated, time: 10 minutes; permissible time values: 1-180 minutes.

6.1.2. Access Right Management

User management in a project is only useful in combination with the access right management.

In a new project, all rights are initially set to a default value, usually *granted*.

Rights can be explicitly granted or denied and reset to default as needed. The access right management of a project is handled in the *Permissions* dialog or, for object access rights, in the *Access control* dialog within the object *Properties* dialog.

Access rights on objects are not inherited. For example, if an action is assigned to a program object in the structure tree, the program is the *father* of the action object. The current rights of the father object do not automatically become the default settings of the child object.

The *Permissions* dialog syntax indicates relationships, such as <*father object*>.<*child object*>. For example, if the *modify* right is denied for UserPrg and a certain user group, the default *modify* right for ACT (a child of UserPrg) is not automatically *denied*.

To access the *Permissions* screen, navigate to Project > User Management > Permissions. This will open the window shown in the figure below.

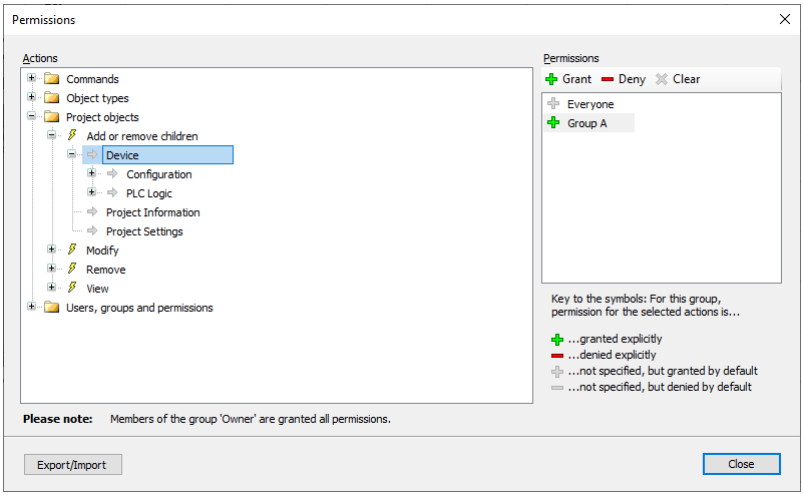

Figure 49: Permissions

The Actions window displays all possible rights within the current project, organized as follows:

- : Top-level categories for optical structuring of rights management, grouping actions related to Commands, User accounts, Groups, Object Types, and Project Objects.
- : Nodes under each category representing specific actions that can be performed.

Possible actions include:

- *Execute* (execution of a menu command).
- *Create* (creating a new object).
- *Add or Remove children* (managing child objects).
- *Modify* (editing an object).
- *Remove* (deleting or cutting an object).
- *View* (viewing an object).
- : Targets for each action.

The Permissions window lists all user groups (except *Owner*) and provides a toolbar for configuring rights.

Icons indicate the currently assigned permissions for the selected target in the Actions window:

- : Action(s) granted for the group.
- : Action(s) denied for the group.
- : Action(s) granted by default.
- : Action(s) denied by default.

To configure rights for a group:

1. Select the desired action(s) in the *Actions* window.
2. Select the desired group in the *Permissions* window.
3. Use the appropriate button in the toolbar:
   - Grant: Explicitly grant the right.
   - Deny: Explicitly deny the right.
   - Clear: Reset to default.

6.2. User and Access Right Management of the CPU

Nexto CPUs have a user rights management system that blocks or allows certain actions for each user group in the CPU. To edit these rights in the CPU, the user must access a project in MasterTool IEC XE without being logged in to the CPU. He must then click in the Device tree on the left of the program, double click on the *Device* item and then select the CPU in the *Communication Settings* tab that opens. Only the *Users and Groups* and *Access Rights* tabs refer to this topic. The figure below illustrates the steps to access this CPU tab.

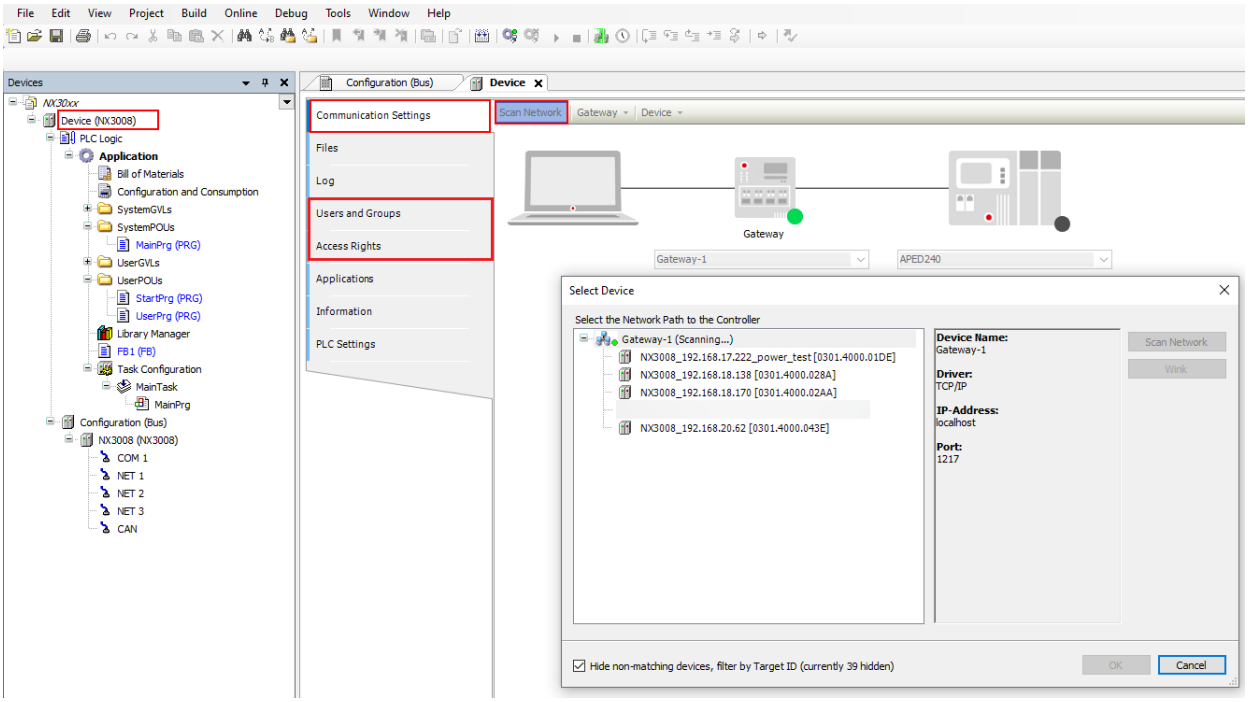

Figure 50: Access to Users and Groups and Access Rights Tabs

ATTENTION

If the user forgets the password(s) of the account(s) with access to the CPU, the only way to recover this access is by updating its firmware.

ATTENTION

After performing a CPU user Logoff command, the *Device* tab of this project must be closed, so that all access rights are effectively closed.

6.2.1. Users and Groups

The *Users and Groups* tab is located in the *Devices* tab. It allows the configuration of user and group accounts that which, together with access rights management, control access to objects in the PLC in online mode.

6.2.1.1. Common

It might be desired that certain functions of a controller can only be executed by authorized users. For this purpose, the *Online User Management* feature provides the possibility to set up user accounts, to assign access rights for user groups and to force an user authentication at login.

The device specific user management might be predefined by the device description and it also depends on this description to what extent the definitions can be edited in the configuration dialogs in the programming system.

Like in the project user management, users have to be members of groups and only user groups can get certain access rights.

6.2.1.2. Using the Configuration Dialog

Basically, the handling of the online user management dialogs is very similar to that of the project's user management. There is even the possibility to *import* user account definitions from the project's user management.

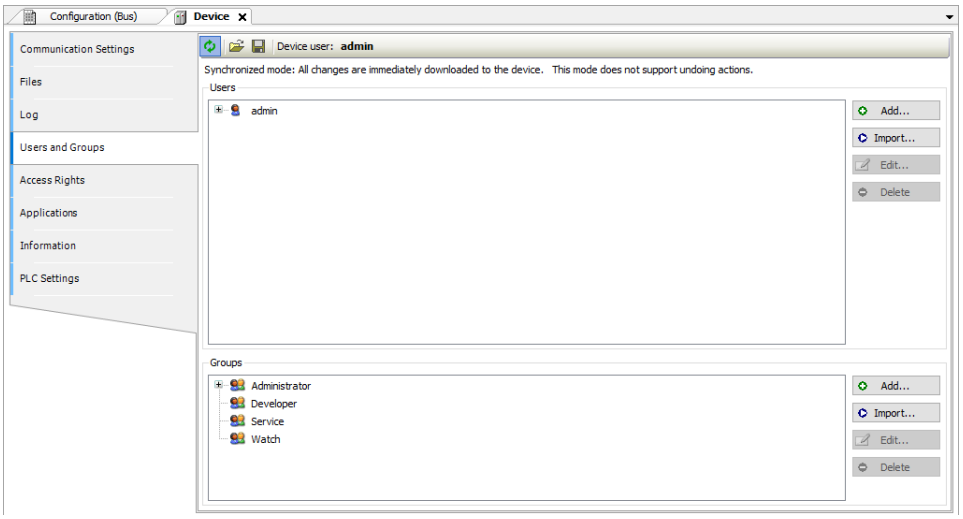

Figure 51: Device Dialog, Users and Groups

6.2.1.2.1. Users

The following buttons are available for setting up user accounts:

Add: The dialog *Add User* opens where you can define a user name and a password. The password must be repeated in the *Confirm password* field.

ATTENTION

By opening this dialog, the *Password* and *Confirm Password* fields are going to be filled with fictional characters, the user must replace these characters with a valid password.

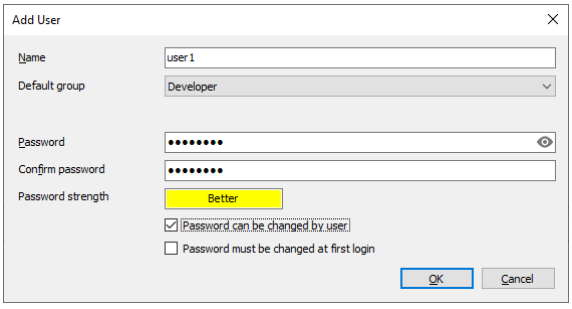

Figure 52: Add User Dialog

Import: The dialog *Import Users* opens showing all user names which are currently defined in the project user management. Select one or several entries and confirm with *OK*. The dialog *Enter* password will open where you have to enter the corresponding password as it is defined in the project user management, in order to get the user account imported to the device-specific user management.

Modify: The currently selected user account can be modified concerning user name and password. This *Edit User* <user name> dialog corresponds to the *Add User* dialog.

Delete: The currently selected user account will be deleted.

6.2.1.2.2. Groups

Add: The dialog *Add Group* opens where you can define a new group name and select from the currently defined users those who should be members of this group.

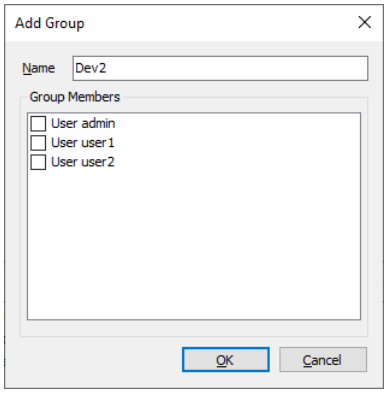

Figure 53: Add Group Dialog

Import: The dialog *Import Groups* opens where the groups currently defined in the project user management are listed. You can select one or several entries and confirm with *OK* to get them integrated in the groups list of the device-specific user management.

Modify: The currently selected group can be modified concerning group name and associated users. For this purpose the *Edit Group* <group name>, dialog opens, which corresponds to the *Add Group* dialog.

Delete: The currently selected group can be deleted.

6.2.1.3. Applying and Storing the Current Configuration

Enables or disables synchronization between the editor and the device's user management system.

When the button is not activated, the editor remains either blank or displays a configuration that you have loaded from the hard disk.

If you enable synchronization while the editor holds a user configuration that hasn't been synced with the device, you'll be prompted to decide how to handle the editor's contents. You have the following options:

- **Upload from the device and overwrite the editor content**: This will load the configuration from the device into the editor, replacing the current contents.
- **Download the editor content to the device and overwrite the user management there**: This will transfer the configuration from the editor to the device, applying it and overwriting the existing user management settings.

Save to Disk, Load from Disk: The current configuration can be stored in a *.dum2 file and re-loaded from this file, which is useful to set up the same user configuration on multiple systems. The standard dialog for browsing in the file system will be provided for this purpose. The file filter automatically is set to *.*dum2*, which means *device user management* files.

Note: Before CODESYS V3.5 SP16, the Device user management files (*.dum) file type was used which did not require any encryption.

The actual settings can also be documented as printed version by use of the command *Print...* (*File* menu) or *Document...* (*Project* menu).

6.2.1.4. Considerations about Default Users and Groups

In firmware versions 1.3.x.x or lower, the existing users and groups are: Everyone and Owner, as seen in the table below:

Users	Groups
Everyone	Everyone
Owner	Owner

Table 18: Users and Groups in Versions 1.3.x.x

However, in firmware versions 1.4.x.x or higher there are the users: Administrator and Everyone, and the groups: Administrator, Developer, Everyone, Service and Watch. As seen in the table below:

Users	Groups
Administrator	Administrator
Everyone	Developer
	Everyone
	Service
	Watch

Table 19: Users and Groups in Versions 1.4.x.x

6.2.1.4.1. Group Administrator

This group has all privileges and it is not possible to remove it in the firmware versions 1.4.x.x or higher. The group Developer is part of this group.

6.2.1.4.2. Group Developer

Group created to define access rights to users that are application developers. The group Service is part of this group. If not used, this group can be removed.

6.2.1.4.3. Group Everyone

For firmware versions 1.3.x.x or lower: This is the default group to perform accesses in a CPU while there are no defined users and groups.

For firmware versions 1.4.x.x or higher: This is the default group to perform accesses in a CPU while there are no defined users and groups.

6.2.1.4.4. Group Service

Group created to define access rights to users that provide some kind of service in the PLC, for example, maintenance teams. The group Watch is part of this group. If not used, this group can be removed.

6.2.1.4.5. Group Watch

Group created to define access rights to user that can only visualize, without making any modification in the application. If not used, this group can be removed.

6.2.1.4.6. User Administrator

The user Administrator is defined in the groups Everyone and Administrator. The password of the user Administrator is *Administrator* and can be modified.

6.2.1.4.7. User Everyone

For firmware versions 1.3.x.x or lower: The user Everyone is defined in the group Everyone. This user doesn't have a defined password.

For firmware versions 1.4.x.x or higher: The user Everyone is defined in the groups Everyone and Administrator. This user doesn't have a defined password.

6.2.1.5. Users and Groups from Old Projects

To maintain this data from old projects in a new project after updating the CPU firmware or in a new Nexto CPU, it's necessary to execute the command *Synchronization* () in the old project with the original firmware, thus fetching the CPU configuration, and then execute the command *Save to disk*, saving the current configuration in a file.

In the new Nexto CPU or in an updated CPU, run the command *Load from disk*, then select the file generated before, *Synchronization* again, thus sending the configuration to the CPU.

ATTENTION

If the old project is with firmware versions 1.3.x.x or lower, a user and a group with the name *Administrator* must be created before saving the configurations in a file. This procedure guarantees that the configuration will be loaded in projects with firmware versions 1.4.x.x or higher.

6.2.2. Access Rights

This dialog is provided on a tab of the *Device* dialog (Device Editor). It is part of the *Online User Management* feature and is used to grant or deny certain permissions to the currently defined user groups, thus defining the user's access rights to files or objects (such as an application) on the PLC at runtime.

Note that these rights can only be assigned to groups, not to individual users. For this reason, a user must be defined as a member of a group. Users and groups are configured in the *Users and Groups* tab of the Device Editor.

The figure below shows the rights to add and remove children to/from the Device object for the user groups *Administrator*, *Developer*, *Service* and *Watch*.

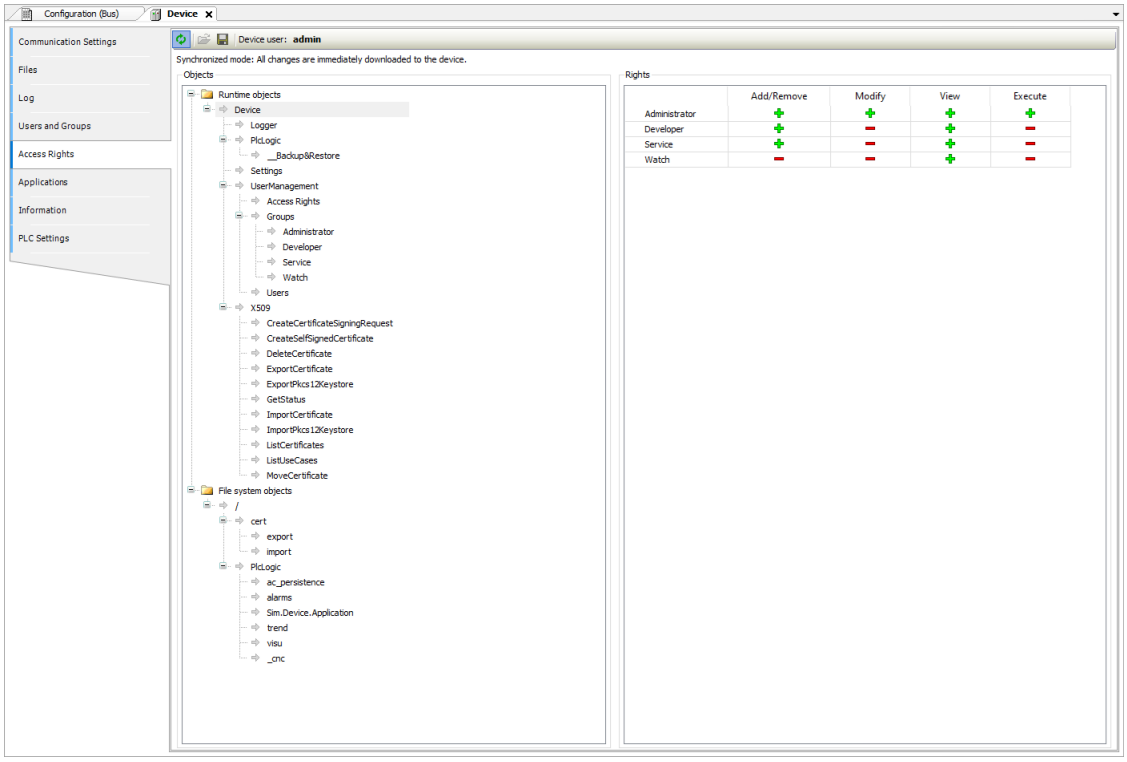

Figure 54: Device Access Rights

See in the following how to define the access permissions and how to get them loaded to the device or stored in a reloadable file.

6.2.2.1. Defining the Access Rights

To define the permission for performing an action on one or multiple object(s), you have to select the object entries below the desired action type in the *Objects* window, then select the desired group in the *Rights* window, and then click on the *Grant* or *Deny* button (also in the *Rights* window).

See in the following a description of the particular dialog windows.

6.2.2.1.1. Objects

This part of the dialog lists the possible actions which might be performed during runtime on files in the PLC file system resp. runtime objects like for example applications. The tree is structured in the following way:

- :
  - Top-level, for structuring purposes, there are two *folders* for File system objects and Runtime objects.
- :
  - Indented below there are nodes for the four types of actions, which might be performed on the particular objects. These nodes also just serve for structural purposes:
  - Add/remove children (adding or removing of *child* objects to an existing object).
  - Execute (for example start/stop application, setting breakpoints etc.)
  - Modify (for example downloading application, etc.)
  - View (monitoring)
- Objects (action *targets*)

ATTENTION

Assigning an access right definition to a *father* in the objects tree usually means that the *children* will inherit this definition, as long as they do not get an explicit own definition. However, depending on the device, this might be handled differently. In any event, inheritances are not visualized here in the dialog.

6.2.2.1.2. Rights

This field shows the defined user groups and their rights in a table. By selecting an Object in the *Objects* tab, you can change its rights using the following buttons:

- : The object(s) currently selected in the *Objects* window are granted for the group.
- : The object(s) currently selected in the *Objects* window are denied for the group.
- : Currently there is no explicit access right definition for the object(s) currently selected in the *Objects* window.

6.2.2.2. Applying and Storing the Current Configuration

Enables or disables synchronization between the editor and the device's user management system.

When the button is not activated, the editor remains either blank or displays a configuration that you have loaded from the hard disk.

If you enable synchronization while the editor holds a user configuration that hasn't been synced with the device, you'll be prompted to decide how to handle the editor's contents. You have the following options:

- **Upload from the device and overwrite the editor content**: This will load the configuration from the device into the editor, replacing the current contents.
- **Download the editor content to the device and overwrite the user management there**: This will transfer the configuration from the editor to the device, applying it and overwriting the existing user management settings.

Save to Disk, Load from Disk: The current configuration can be stored in a *.drm file and re-loaded from this file, which is useful to set up the same user configuration on multiple systems. The standard dialog for browsing in the file system will be provided for this purpose. The file filter automatically is set to *.*drm*, which means *device rights management* files.

The actual settings can also be documented as printed version by use of the command *Print...* (*File* menu) or *Document...* (*Project* menu).

6.2.2.3. Access Rights of Old Projects

To maintain this data from old projects in a new project after updating the CPU firmware or in a new Nexto CPU, it's necessary to execute the command *Synchronization* ( ) in the old project with the original firmware, thus fetching the CPU configuration, and then execute the command *Save to disk*, saving the current configuration in a file.

In the new Nexto CPU or in an updated CPU, run the command *Load from disk*, then select the file generated before, *Synchronization* again, thus sending the configuration to the CPU.

ATTENTION

If the old project is with firmware versions 1.3.x.x or lower, a user and a group with the name *Administrator* must be created before saving the configurations in a file. This procedure guarantees that the configuration will be loaded in projects with firmware versions 1.4.x.x or higher.

7. Menu Commands

The available commands for MasterTool IEC XE user interface are standard. See Standard Menus and Commands to check the standard menu structure.

7.1. File Menu

The *File* menu provides commands, which can be used for handling a project file.

- New Project
- Open Project
- Close Project
- Save Project
- Save Project As
- Project Archive
  - Extract Archive..
  - Save/Send Archive..
- Source Upload
- Source Download
- Print
- Print Preview..
- Page Setup
- Recent Projects
- Exit

7.1.1. New Project

Symbol: 

Default Shortcut: <CTRL>+<N>

This command is used to create a new project with the aid of dialog *New Project*.

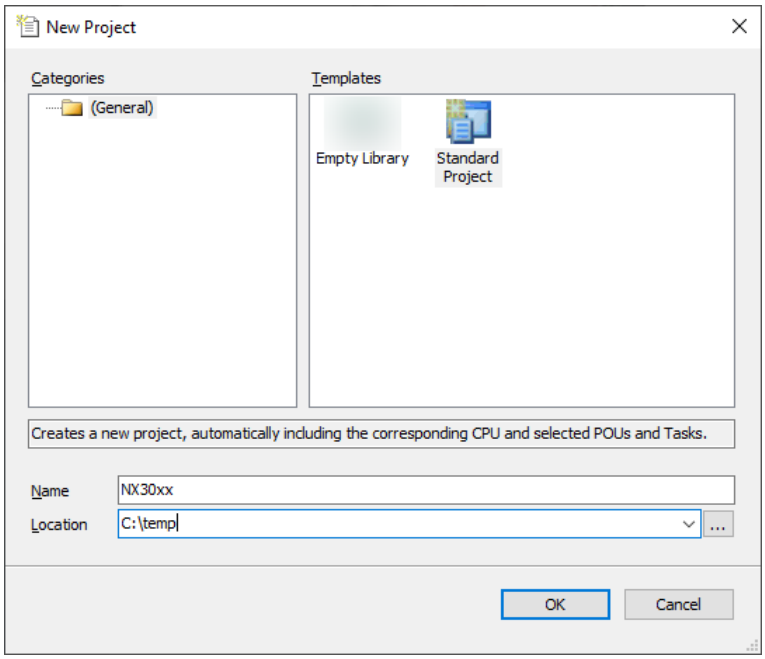

Figure 55: Project Classification

1. *Categories*: The available categories of templates and project wizards are offered in a tree structure. If a category is selected the assigned templates and wizards will be displayed in the Templates list on the right side.
2. *Templates*: This is a list of all templates/wizards belonging to the category currently selected in the Categories tree on the left side. A template determines the base configuration of a project file. There is a model Standard Project (.project) that automatically provides the CPU and inserted devices, POUs and tasks according to user choices. There is also a model for Empty Library (.library).
3. *Name*: Name of the project to be created. The default name is specified by the currently selected template/wizard and includes a numeric suffix guaranteeing the uniqueness within the file system (for example *Untitled1*). You can edit the entry considering the file path conventions of the operating system. It should not be used for special characters and should be subject to the maximum limit of 200 characters. Optionally you can add an extension (for example *.project*), by default automatically the extension defined by the currently selected template/wizard will be assigned.
4. *Location*: Location of the new project file. The default path is specified by the currently selected template/wizard. You can use the Windows standard browser for modifying the path via ... button or choose one of the recently used location paths via button.
5. *OK*: A new project will be created according to the done settings. If any settings are missing, an error icon ()will be displayed at the respective entry field in the dialog. When the cursor is placed on an error icon, a tooltip will provide a hint what to do. If a project was already opened when setting up the new one, you will now be asked whether this should be saved and closed before creating the new project.

The name of the new project in each case will be displayed in the title bar of the frame window of the programming system.

Note: An asterisk behind the project name will indicate that the project has been modified since the last save.

7.1.2. Open Project

Symbol:

Default Shortcut: <CTRL>+<O>

This command can be used to open an existing project file with the aid of the standard dialog for opening a file.

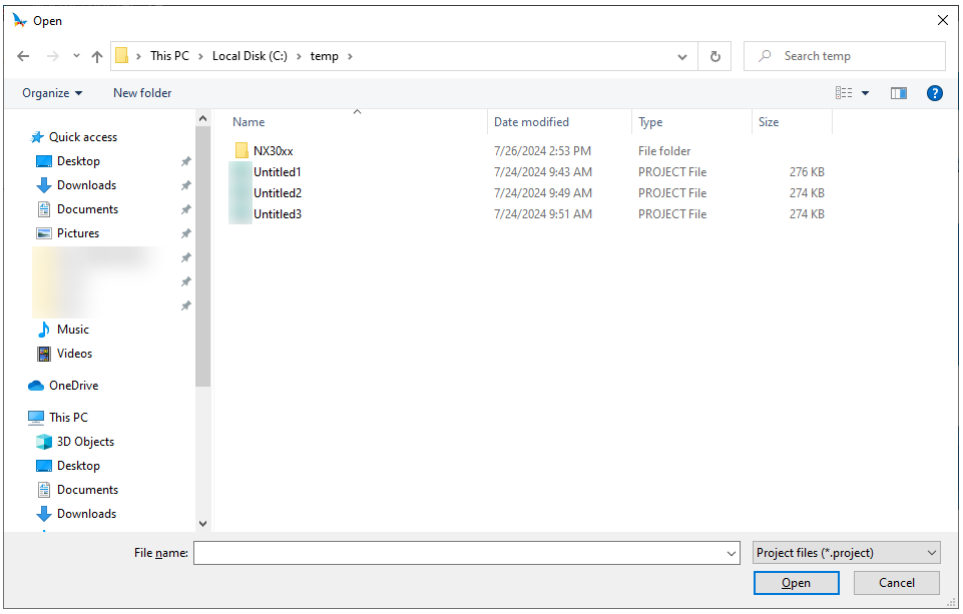

Figure 56: Open Project

When you select Open, the chosen project will be opened or not. The following cases are possible:

- Another project is still open
- Project was not terminated regularly and *AutoSave* was activated
- Project is read-only

7.1.2.1. Another Project is Open

You will be asked whether it should be saved and closed

7.1.2.2. Project Not Terminates Regularly (Auto-save activated)

If the *Auto Save* function had been activated and MasterTool IEC XE had been terminated non-regularly before the last project was saved after a modification, you will get the *Auto Save Backup* dialog when going to re-open the same project. For details see Load and Save.

7.1.2.3. Project is Read Only

If the project you want to open is read-only on disk, you will be asked whether you want to open the project in read-only mode or whether you want to make the project writable.

7.1.3. Close Project

This command will close the project while the programming system will stay opened. If the project contains non-saved modifications, you will be asked whether you want to save these changes.

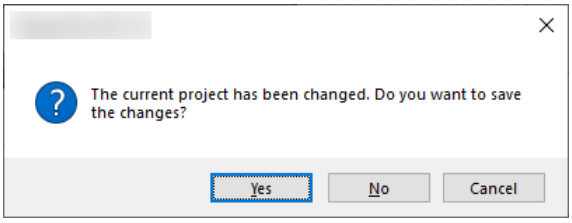

Figure 57: Project Alterations

For terminating MasterTool IEC XE use command *Exit*.

7.1.4. Save Project

Symbol:  **Default Shortcut**: <CTRL>+<S>

This command is used to save the project at the currently defined location. It is only available if any modifications have been done in the project since the last saving. This is indicated by an asterisk behind the project name in the title bar.

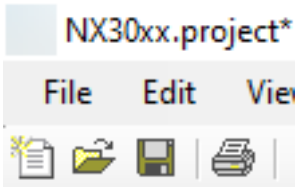

Figure 58: Title bar, Modified Project

In case the project is write-protected, the following message dialog will be opened:

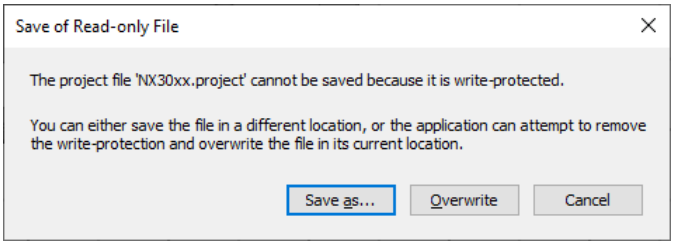

Figure 59: Message Dialog, Project Write-Protected

The *Save As...* field allows to define a new project path.

7.1.5. Save Project As

This command is used to save the project, whereby location and type of the file can be defined. The Windows standard dialog for saving a file will be opened for this purpose.

In the *Name* field, the file name is already set. If desired browse for another path and/or edit the file name entry.

In the *Data type* field choose one of the available file types. Regard the following before saving that project as a library:

- If the library should later be suitable for getting installed in other projects, enter at least a title and a version number, optionally (recommended) also a category and the company name in the Project Information (Summary).
- The saving of a library project does not include an automatic check for errors.

If there is already a file with the defined name, a message dialog will open to ask you whether that file should be replaced. If you choose *No*, you will get back to the *Save Project* dialog described above to select another path.

7.1.6. Project Archive

7.1.6.1. Extract Archive...

This command is used to extract an archive file (default file extension *.projectarchive*) that has been created by use of the command *Save/Send Archive*.

Regard that extracting an archive will require to close all currently loaded projects in any of the currently opened instances of the programming system. This is to avoid direct impact on running projects if for example libraries are changed due to the extract operation.

The archive file can be selected via the standard dialog box that arises in response to the command execution. After confirmation of the selection with *Open* there will pop up a dialog box.

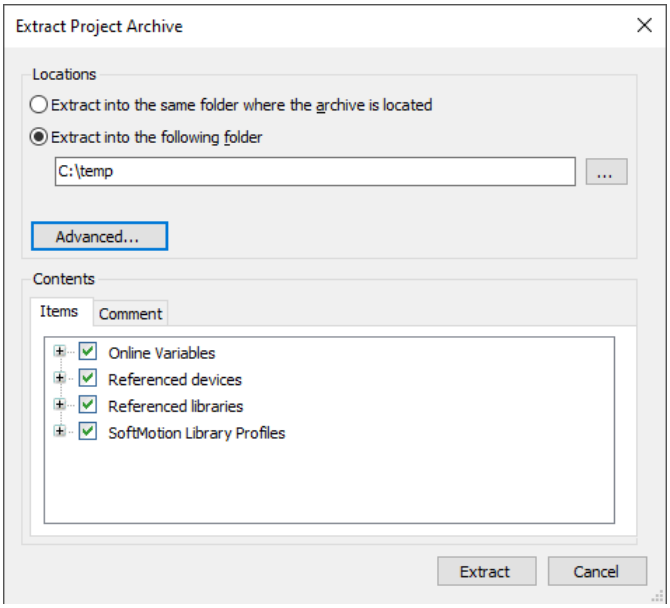

Figure 60: Extract Project Archive

Under *Extract into the following folder*: the path to which the archive is to be extracted to is specified. You can modify the target directory either by hand (click on the path indication and modify it by typing) or by a click on ..., what gives rise to the standard dialog for browsing.

The lower rectangle of the dialog box displays the file categories packed in the archive. A click on the sign, which antecedes each category, will expand a list of its associated files. All categories mentioned are marked by ☑, what indicates that all of the corresponding files will be extracted. If you want to prevent certain files or even an entire category from being extracted, you have to deactivate their marking by a click (the mark will then appear as for partial deactivation ■, respectively for entire deactivation ☐).

The archive may contain other than the project files that have been added explicitly. After a click on *Advanced* there will pop up the following dialog box:

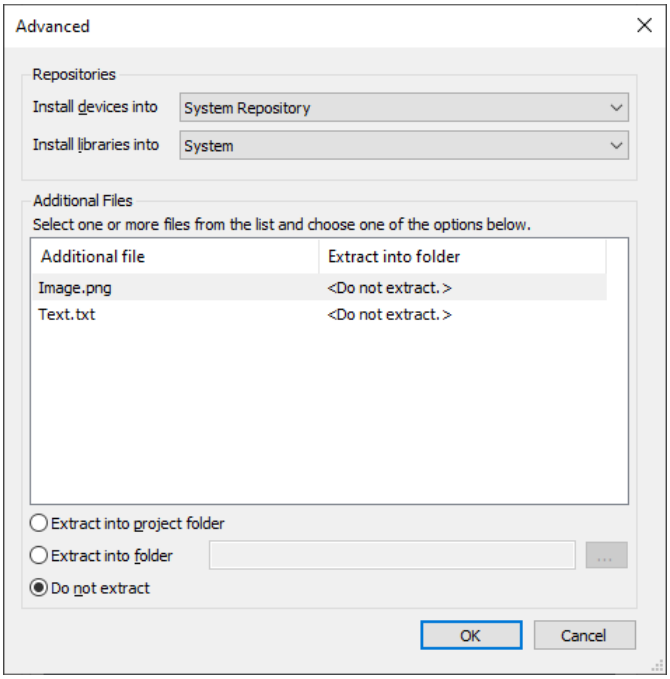

Figure 61: Project Archive, Advanced

By default, an additional file will not be extracted together with the project files (see the remark <Do not extract> to the right of the file name). If you want an additional file to be extracted, you have to select the file and choose one of the following options:

- *Extract into the project folder*: will extract the selected file to the same directory as the project files specified by the path. The remark to the right of the file name will be changed into <Extract to the project folder.>.
- *Extract into folder*: will extract the file to the folder that you can specify either by typing the corresponding path into the text field or by making use of the standard dialog for browsing after a click on .... The remark to the right of the file name will be changed to the specified path.
- *Do not extract*: will reset the selected file to the default mode.

A click on *Comment* will display the comment eventually given on the archived project by its author. If no comment is contained in the archive, a warning message will be displayed in this place.

Having configured the set-up in the dialog box you may click on: to unpack the archive to the specified path:

- *Extract*: If a file to be extracted has the same name as a file that already exists in the destination directory, a warning message will be displayed and you'll be asked whether you want to replace the local file or not. You can make your decision take effect for each subsequent extraction.
- *Cancel*: to abort operation; the archive will not get extracted.

7.1.6.1.1. Extracting Files from Projects Created in Older Versions

In many situations, when a project is created in a MasterTool IEC XE version, and it's then opened in a later version, it's necessary to run a *Project Update* command. This project update, in most cases, will cause the project to be changed and therefore make it no longer possible to execute a login command in the CPU. Probably it won't also be possible to execute an *Online Change*. To avoid this kind of situation it's possible to create a Project Archive (*.projectarchive) with the MasterTool IEC XE older version. This project file can be opened in the new version of MasterTool IEC XE and so the login can be done.

In some situations, it's possible that the old project files also cannot be extracted in a new MasterTool IEC XE version, generating compilation errors and making it necessary to do a *Project Update*. In MasterTool IEC XE version 2.00, many features were changed. When a project archive is created in a MasterTool IEC XE version prior to 2.00 and the opened in version 2.00 or higher, some compilation errors can occur, depending on the features used on the project. In these cases, it will be mandatory to carry out a project update.

7.1.6.2. Save/Send Archive...

This command is used to set up and create a project archive file, and all files, which are referenced by and used within the currently opened project, are packed. The archive file (<filename>.projectarchive) can either be stored or sent as attachment of an email. The latter is useful to forward the set of all project-relevant files as the archive can be unpacked easily by using the command *Extract Archive*.

Note: The archive function is not intended for restoring a project environment. It is designed for easy packing of all files belonging to a project.

Executing the Save/Send Archive (menu *File* > *Project Archive* > *Save/Send Archive...* > *Additional Files*), the following dialog box will open.

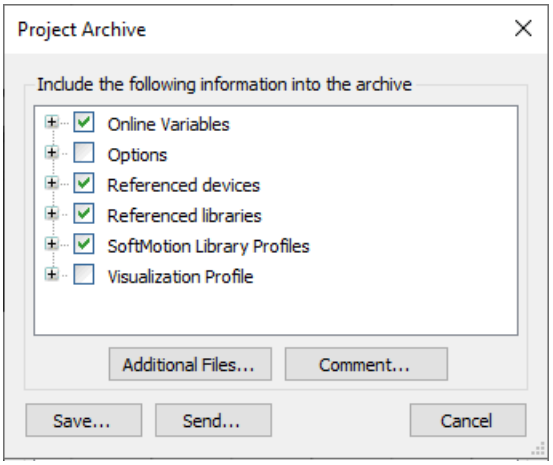

Figure 62: Project Archive Dialog

In this dialog box you can define which file categories should be added to the archive file: Select or deselect a category by activating/deactivating the corresponding checkbox. Do this by a single mouse click in the checkbox or by a double click on the category name. If a category is marked by ☑, all files belonging to this category will be added to the archive file. A category is marked by ☐ if the project does not include any corresponding file.

To select/deselect a single file of a special category press the aligned button ⊞ to get a list of all associated files. By default, all files of an activated category are selected.

To modify a selection activate or deactivate the desired files in the same way as the categories. Now, if not all but only certain files of a category are activated, its mark will appear as ▣.

To add any other files to the archive than the ones listed above, press the button *Additional files...*, and the corresponding dialog box opens.

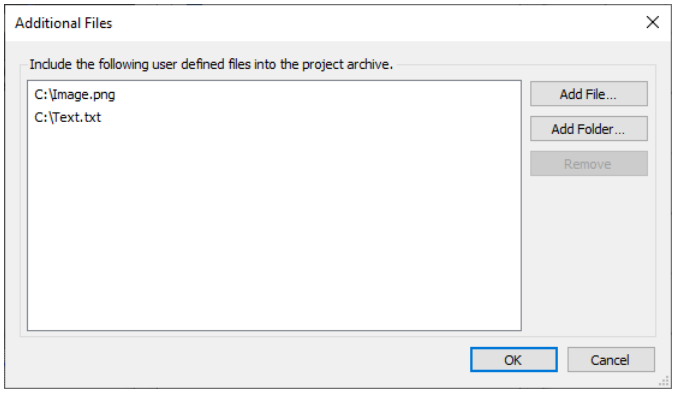

Figure 63: Additional Files Dialog

Press the button *Add File...* to open the standard dialog, where you can browse for files (sorted by file type) and open the ones you selected (by confirmation with *Open*). The selected files will be added to the list of the *Advanced* dialog. There's also the button *Add Folder...* to add an entire folder and the content inside it. To remove a file or folder from this list, select it by a click and press the button *Remove*. After the list of selected files fits your demands, close the dialog with *OK*.

To add a note on the project press the button *Comment...* you may enter your comment in the rising text editor before closing it with *OK*. When your archived project is imported, your comment can be accessed in the corresponding dialog via *Comment*.

- *Save*: to create and store the archive file to a desired path that you select via the opening standard dialog. Therein, you may also change the default name <projectname>.projectarchive of the archive file. Confirm with Save to start the building process.
- *Send*: To create a temporary archive file that will be attached to a simultaneously generated empty email. The successful operation of this feature relies on a correct installation of an e-mail client. If the operation has been unsuccessful, an error message will be displayed.
- *Cancel*: To cancel the action; no archive file is generated.

7.1.7. Source Upload

This command is available in the *File* menu for opening a project from a PLC. For this purpose, a project archive file must be available there, possibly generated by the *Source download* function.

The command opens the *Select Device* dialog, where you have to choose the network path to the PLC like in the *Communication Settings* dialog. Select the appropriate entry in the tree of available devices and press *OK*.

The *Project Archive* dialog opens, where you can configure which contents of the archive should be extracted for upload and to which path they should be copied. The usage of this dialog corresponds to that of the *Project Archive/Extract archive* function. After confirmation with *OK* the files will be copied. If a file is already available in the specified path, you will be asked whether it should be overwritten.

Then a dialog box will appear, asking whether the extracted project should be opened in the programming system.

7.1.8. Source Download

This command is available for creating and transferring an archive file of the actual project to any device.

The command opens the *Select Device* dialog, where you have to choose the network path to the PLC like in the *Communication Settings* dialog. Select the appropriate entry in the tree of available devices and press *OK*.

This will set up a connection to the device as long as the source code gets downloaded in the form of an archive file.

The source code can be re-load to the programming system in offline mode by using command *Source upload*.

The default settings concerning destination device, content and timing for the source download are defined in the *Project Settings*, category *Source Download*.

7.1.9. Print

The currently active editor view can be printed by using the *Print* commands.

7.1.10. Print Preview...

The command opens a print preview of the for the element which is The command opens a print preview of the element which is currently open.

7.1.11. Page Setup

Configures the printout layout. For further information, see Page Setup.

7.1.12. Recent Projects

Use this command to select from a list of the most recently opened projects that one you want to reopen.

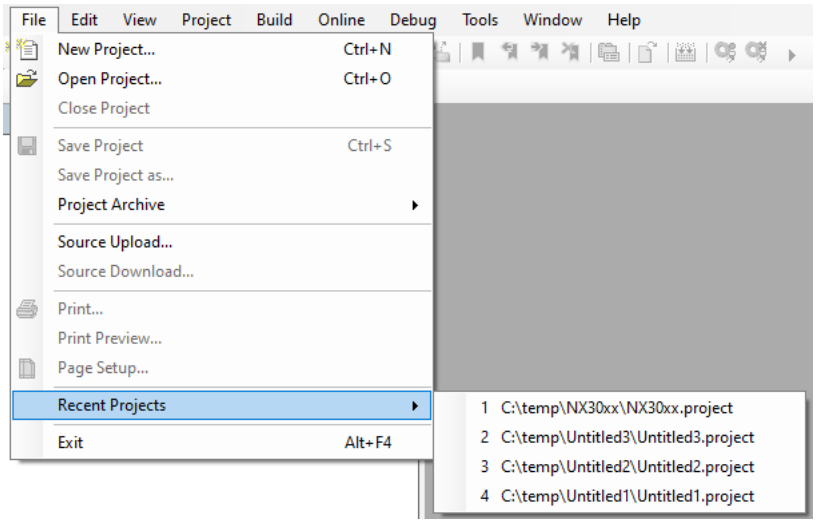

Figure 64: Recent Projects List

7.1.13. Exit

Default Shortcut: <ALT>+<F4>

This command will terminate the programming system. If currently a project is opened which has been modified since the last saving, a dialog will open asking you whether the project should be saved.

7.2. Edit Menu

This menu provides commands for device and objects editing in its respective editors.

- Undo
- Redo
- Cut
- Copy
- Paste
- Delete
- Select All
- Find/Replace
  - Find
  - Replace
  - Find in Project
  - Replace in Project

- Find Next
- Find Next (Selected)
- Find Previous
- Find Previous (Selected)

- Browse
  - Go to Definition
  - Display Cross References
  - Display Call Tree
- Insert File as Text
- Advanced
  - Overwrite Mode
  - Go to Line
  - Make Lowercase
  - Make Uppercase
  - Go to Matching Bracket
  - Select to Matching Bracket
- Bookmarks
  - Toggle Bookmark
  - Next Bookmark
  - Previous Bookmark
  - Clear All Bookmarks
- Input Assistant
- Auto Declare..
- Next Message
- Previous Message
- Go to Source Position
- Refactoring
  - Rename '<Var>'..
  - Add Variable..
  - Remove '<Var>'..
  - Reorder Variables..
  - Update Referenced Pins

7.2.1. Undo/Redo

Provides the following commands for restoring previous steps during editing an object in a project.

Available commands:

- Undo
- Redo

7.2.1.1. Undo

Symbol: 

Default Shortcut: <CTRL>+<Z>

This command undoes the action, which was most recently executed in the currently open editor or view window.

Repeated use undoes all actions back to the time that the window was opened. This applies to all actions in the editors for POUs, data types, visualizations and global variables.

With the *Redo* command, the user can restore an action, which you have undone before.

7.2.1.2. Redo

Symbol: 

Default Shortcut: <CTRL>+<Y>

With this command in the currently opened editor or view window an action can be restored, which has been undone (*Undo*) before.

7.2.2. Clipboard

The command category *Clipboard* provides the usual functions to manage contents between the project and the clipboard.

- Cut
- Copy
- Paste
- Delete

7.2.2.1. Cut

Symbol: 

Default Shortcut: <CTRL>+<X> or <SHIFT>+<DELETE>

This command transfers the current selection (object entry, string) to the clipboard. The selection is removed from the editor and object tree. In tree structures, which are used to organize objects, as for example in the POUs view, this applies to the selected object. Multiple selection is possible.

Remember that not all editors support the *Cut* command, and that its use can be limited in some editors.

The form of the selection depends upon the respective editor: For example it is a string or character in text editor's resp. might be one or several elements surrounded by a selection frame in graphic editors.

In order to paste the content of the clipboard you use the *Paste* command.

In order to copy a selection onto the clipboard without deleting it, use the *Copy* command.

In order to remove a selected area without changing the clipboard, use the *Delete* command.

7.2.2.2. Copy

Symbol: 

Default Shortcut: <CTRL>+<C> or <CTRL>+<INS>

This command copies the current selection to the clipboard. This does not change the contents of the editor window. In tree structures, as for example in the POUs view, this applies to the selected object. Multiple selection is possible.

Remember that not all editors support the *Copy* command, and that its use can be limited in some editors.

For this selection type, the same is true as for the *Cut* command.

In order to paste the content of the clipboard you use the *Paste* command.

In order to delete a selected area and simultaneously put it on the clipboard, use the *Cut* command.

7.2.2.3. Paste

Symbol: 

Default Shortcut: <CTRL>+<V> or <SHIFT>+<INS>

This command pastes the content of the clipboard onto the current position in the editor window.

Pasting is not supported by all editors and its use might be limited. In graphic editors the command is only supported if a correct structure will result from the insertion.

Multiple selection is possible. Depending on the current position, for example in the devices tree, a dialog might be opened where you have to choose, whether the object from the clipboard should be entered below or above.

In order to copy a selection onto the clipboard without deleting it, use the *Copy* command.

In order to remove a selected area without changing the clipboard, use the *Delete* command.

7.2.2.4. Delete

Symbol:

Default Shortcut: <DEL>

This command deletes the selected area from the editor window. It does not change the contents of the clipboard.

The command applies to the selected object.

For the type of selection, the same rules apply as with the *Cut* command.

In order to delete a selected area and simultaneously put it on the clipboard, use the *Cut* command.

7.2.3. Select All

Default Shortcut: <CTRL>+<A>

This function selects all the content of the currently opened device. For example, in POUs and lists, it selects the complete code. In the graphic editor, it selects all the devices that are there.

Remember that not all editors support the *Select All* command, and that its use can be limited in some editors.

7.2.4. Find/Replace

The category *Find Replace* provides commands, which can be used to perform a find action concerning certain strings in the project.

- Find
- Replace
- Find in Project
- Replace in Project
- Find Next
- Find Next (Selected)
- Find Previous
- Find Previous (Selected)

7.2.4.1. Find

Symbol:

Default Shortcut: <CTRL>+<F>

Use this command to search the project for a certain string. All editable places within the project objects will be searched.

The *Find* dialog will be opened, where you define which string should be searched according to certain rules, where it should be searched and whether it the found locations should be displayed one after the other or all on a whole. In addition, the user can switch to the Replace dialog.

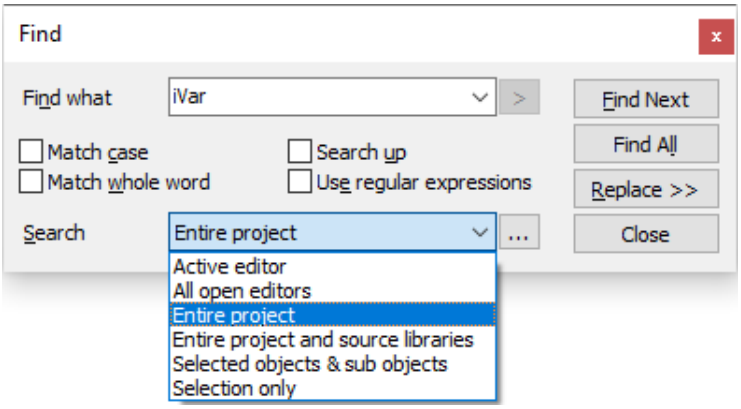

Figure 65: Find Dialog

Find what: Enter the string to search for here. The selection list available via the button ( ) will be filled with strings that have already been searched for since the last time the program was started.

Activate the desired search options:

- *Match case*: The search is case-sensitive referring to the search string.
- *Match whole word*: Only strings, which match the whole search string, will be found.
- *Search up*: The specified search area will be passed upwards. Deactivate the option to search downwards again.
- *Use regular expressions*: Regular Expressions (RegExp), the pattern matching standard for string parsing and replacement, is supported concerning the most commonly used expressions. Use the ( ) button to get assistance for entering the desired combination of those expressions in order to define, which strings and characters should be found. The available expressions are sorted in the following submenus: Special characters, Repetitions, Alternatives, Groups and Others.
- *Search*: Specify here in which objects should be searched for the given string. For this purpose either choose one of the options offered in the selection list via the button ( ), or open the *Search* dialog via button ( ).

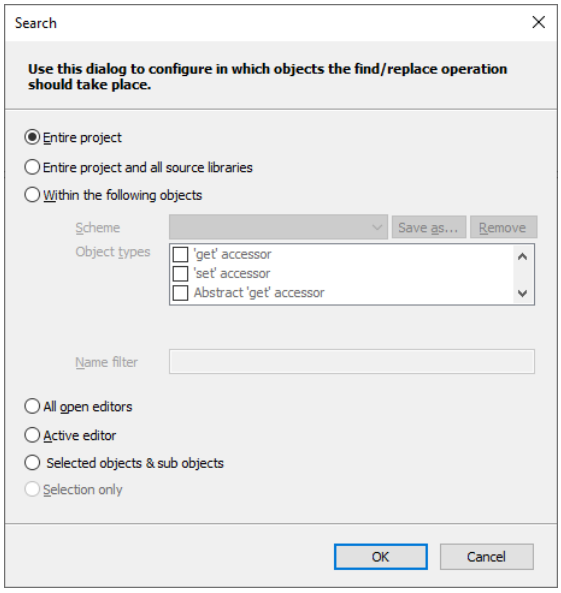

Figure 66: Search Dialog

- *Entire project*: All editable places within all project objects will be noticed.
- *Entire project and all source libraries*: All editable places, within all project objects and all integrated uncompiled libraries will be noticed.
- *Within the following objects*: Only the editable places within those objects will be noticed which are defined by the following settings.
  - *Object types*: Put a check to all object types, which should be searched for.
  - *Name filter*: Optionally set a filter on certain objects names by using placeholders *. Example: Enter **PROFIBUS** to explicitly search for the specified search string in all objects including PROFIBUS in the object name.
  - *Scheme*: Optionally save the currently defined search configuration. Make sure to have set the desired Object types and optionally a Name filter. Then press button *Save as...* and in dialog *Save Scheme* define a name for the current configuration. All saved schemes will be available later in the selection list via button ( ). They can be removed from there via button *Remove*.

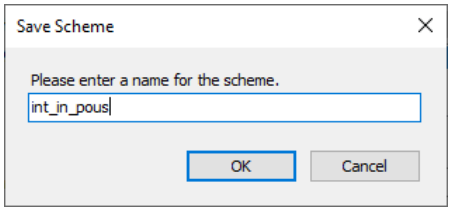

Figure 67: Save Scheme

- *All open editor*: All editors currently opened in a window will be searched.
- *Active editor*: Just the editor where currently the cursor is placed, will be searched.
- *Selected objects & sub objects*: It will search only in objects and sub objects of the program.
- *Selection only*: Only the currently selected text will be searched for the specified search string.

After having set all find and search options press button:

- *Find Next*: To step through the found locations of the searched string step by step. The respective editor windows will be opened and the found string will be highlighted.
- *Find All*: To get a list of the found locations in the Message window. The progress of the search process is displayed within the status line; the search may be interrupted by making use of the button *Cancel* in the status line.

The search being completed the following information is displayed for each location:

- *Description*: Expression containing the search string.
- *Project*: Project name.
- *Object*: Object name.
- *Position*: Position (for example Line number) within the object, in brackets *Decl* for Declaration part resp. *Impl* for Implementation part of the editor window.

Below see the number of total found objects, of matching objects and total objects searched.

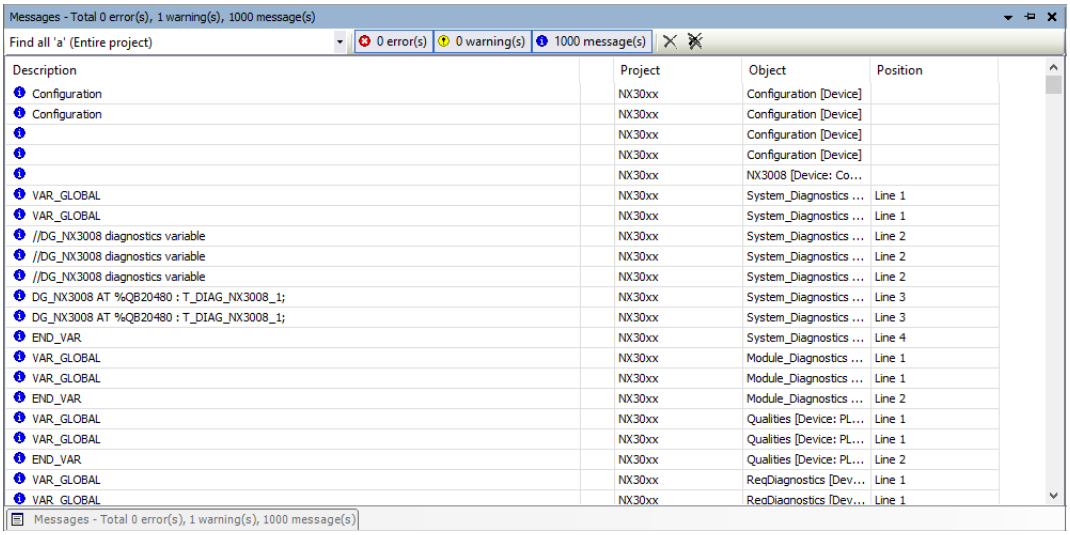

Figure 68: Search Results for String *a*

If you would like to replace the found string by another one, press button *Replace* to get to the Replace dialog.

7.2.4.2. Replace

Symbol: 

Default Shortcut: <CTRL>+<H>

This command opens the *Replace* dialog, which is an extended *Find* dialog.

Like in the *Find*-dialog first set the options for searching for the string, which should be replaced by another one. Additionally enter the new string in the field at *Replace* and then use one of the following replace-buttons:

- *Replace*: Press this button to perform a replacing of the first string, which was found. In this case you can step to the next found string by button *Find Next*.
- *Replace All*: Press this button, if you want to replace all found strings at once.

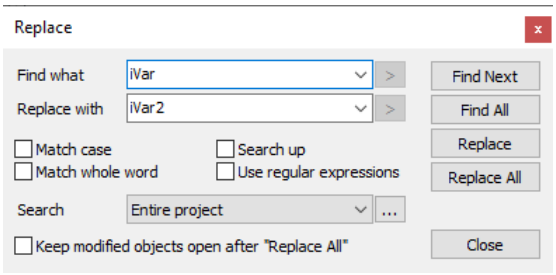

Figure 69: Replace

7.2.4.3. Find in Project

Symbol: 

Default Shortcut: <CTRL>+<SHIFT>+<F>

This command opens the same dialog as the *Find* command. See Find for more information.

7.2.4.4. Replace in Project

Symbol: 

Default Shortcut: <CTRL+<SHIFT>+<H>

This command opens the same dialog as the *Replace* command. See Replace for more information.

7.2.4.5. Find Next

Default Shortcut: <F3>

This command is available to get to the next found position after the *Find* or *Replace* command has been used to search for a certain string. By default, it is part of the *Find & Replace* menu within the *Edit* menu.

7.2.4.6. Find Next (Selected)

Default Shortcut: <CTRL>+<F3>

This command searches for the next string, which matches that one which is currently selected in the editor.

7.2.4.7. Find Previous

Symbol: 

Default Shortcut: <SHIFT>+<F3>

This command is available to get to the previous found position after the *Find* or *Replace* command has been used to search for a certain string. By default, it is part of the *Find & Replace* menu within the *Edit* menu.

7.2.4.8. Find Previous (Selected)

Default Shortcut: <CTRL>+<SHIFT>+<F3>

This command searches for the next string, which matches that one which is currently selected in the editor

7.2.5. Browse

Provides commands for the list of variable cross-references and POU or variable definition. The commands are available in the *Browse* menu.

Available commands:

- Go to Definition
- Display Cross References
- Display Call Tree

7.2.5.1. Go to Definition

Symbol: 

This command can be used when the cursor is currently positioned on an identifier in an editor window. It will browse the project for the line or POU, which contains the definition of the corresponding POU or variable and will open the respective POU in an editor window.

Examples:

The following POU contains a function block definition (*fbinst*), a program call (*prog_y*) and a function block call (*fbinst.out*):

```
VAR
  fbinst:fb1;
  ivar:INT;
END_VAR

prog_y();
ivar:=prog_y.y;
res1:=fbinst.out;
```

If you put the cursor on *prog_y*, the command will open program *prog_y* in its editor window.

If you put the cursor on *fbinst*, the command will set the focus up to the declaration window to the line *fbinst:fb1;*.

If you put the cursor on *out*, the command will open function block *fb1* in its editor window.

7.2.5.2. Display Cross References

Symbol: 

This command should be used in text editors. After selecting a definition and use this command, the *Cross Reference List* window is opened and all the references related to the definition are displayed.

7.2.5.3. Display Call Tree

Symbol: 

This command should be used in text editors. After selecting a definition and use this command, the *Call Tree* window is opened and all the references related to the definition are displayed.

7.2.6. Insert File as Text

This command can be used to insert the content of a text file to the currently opened text editor. The standard dialog for browsing for a file (*Insert File*) will be opened where you can search for the desired file, which must be in text format. The file contents will be inserted at the current cursor position.

7.2.7. Advanced

Depending on the currently active editor, usually text editors, these commands are available in *Edit* menu.

Available commands:

- Overwrite Mode
- Go to Line
- Make Lowercase
- Make Uppercase
- Go to Matching Bracket
- Select to Matching Bracket

7.2.7.1. Overwrite Mode

Default Shortcut: <INS>

Use this command to toggle between Overwrite mode (option activated) and Insert mode (option deactivated. When editing in overwrite mode the existing characters will be overwritten, otherwise the new characters will be inserted.

7.2.7.2. Go to Line

Use this command to jump to a certain line within a text editor. A dialog (Go to Line) will be opened where you can insert the desired line number. After closing the dialog with *OK*, the cursor will be set to the start of the corresponding line.

7.2.7.3. Make Lowercase

Default Shortcut: <CTRL>+<U>

This command will set the currently marked text to lowercase.

7.2.7.4. Make Uppercase

Default Shortcut: <CTRL>+<SHIFT>+<U>

This command will set the currently marked text to uppercase.

7.2.7.5. Go to Matching Bracket

This command will set the cursor at the next matching bracket. This is valid for brackets in program lines as well as for bracket scopes.

7.2.7.6. Select to Matching Bracket

This command will select the code lines up to the next matching bracket. This is valid for brackets in program lines as well as for bracket scopes.

7.2.8. Bookmarks

Menu and sub-menus *Bookmarks* are displayed on the *Edit* menu, depending on the active editor, usually textual editors. Bookmarks can be assigned to one or multiple lines in an editor to make the navigating in long programs easier. Via the appropriate commands, the user can jump to the next or previous bookmark.

The commands:

- Toggle Bookmark
- Next Bookmark
- Previous Bookmark
- Clear All Bookmarks

7.2.8.1. Toggle Bookmark

Symbol: 

Default Shortcut: <CTRL>+<F12>

This command is used in a text editor to set a bookmark in the current line respectively to remove a set bookmark. A dark flag at the left margin will indicate that a bookmark is set.

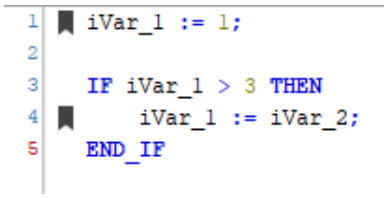

Figure 70: Bookmarks in ST Editor

7.2.8.2. Next Bookmark

Symbol: 

Default Shortcut: <F12>

This command is used in a text editor to jump to the next bookmark.

7.2.8.3. Previous Bookmark

Symbol: 

Default Shortcut: <SHIFT>+<F12>

This command is used in a text editor to jump to the previous bookmark.

7.2.8.4. Clear All Bookmarks

Symbol: 

This command is used to clear all bookmarks in the current editor window.

7.2.9. Input Assistant

Symbol: 

Default Shortcut: <F2>

The *Input Assistant* dialog and the command *Input Assistant* will only be available if the cursor is placed in a text editor window. The dialog offers all project items available for being inserted at the current cursor location.

In the *Text Search* tab it's possible to search for a specific item (figure below). By typing one or more characters in the search field, all items that contain the searched text are going to be listed. It's possible to restrain the search to a variable category through the *Filter* field. With a double-click over the item, it's going to be inserted in the current cursor position in the editor.

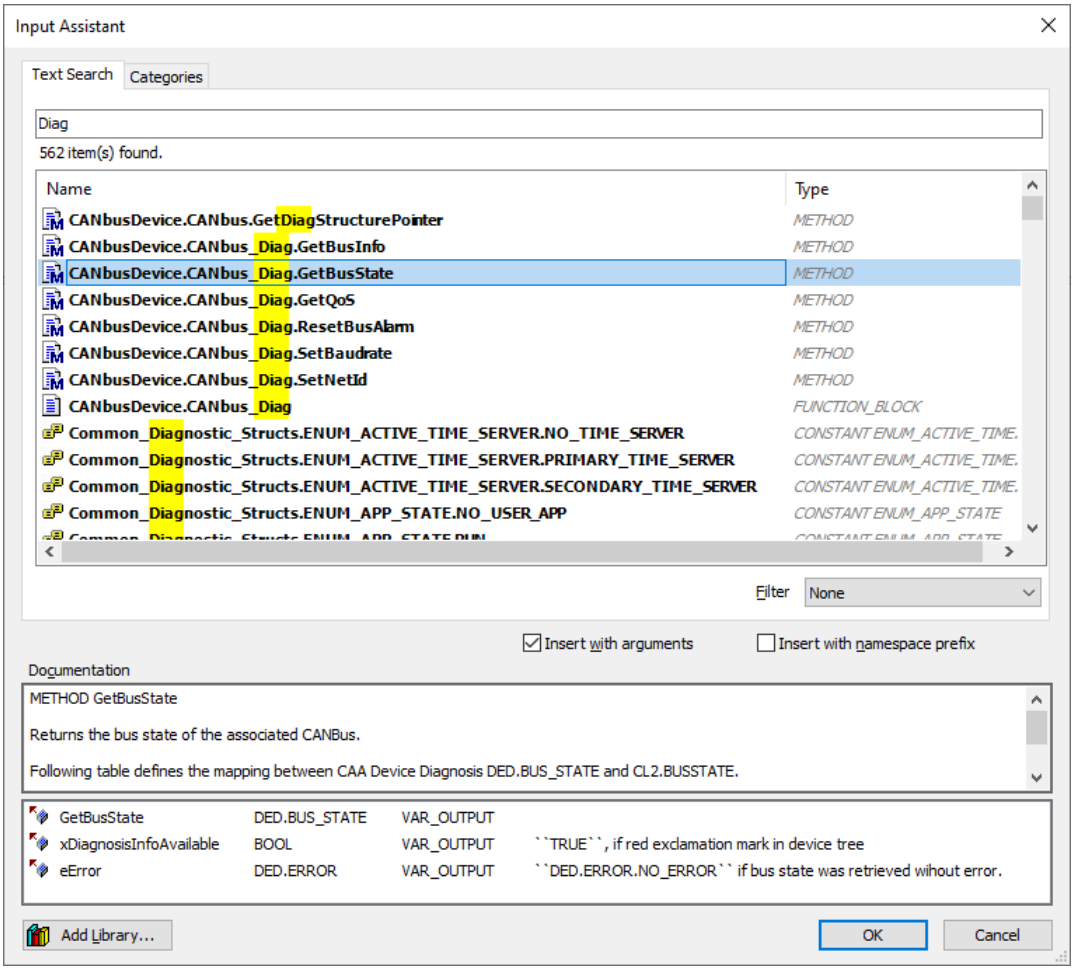

Figure 71: Text Search Tab

In the *Categories* tab, the items are sorted by Categories.

For the currently selected category, the available items and their respective data type are displayed in the field to the left of the screen. If option *Structured View* is activated, the items will be displayed in a structure tree supplemented with icons, otherwise they will be arranged *flat*, but each showing the POU belonging to (for example *GVL1.gvar1*).

Notes:

- If there are objects with the same name available in the global area (POUs tree) as well as below an application (device tree), only one entry will be offered in the *Input Assistant* window, because the usage of the object is determined by the usual call priorities (first the application-assigned object, then the global one).
- The variables shown in the *IoConfig_Globals*, *IoConfig_Application_Mappings* and *IoConfig_Global_Mappings* objects are used internally for I/O control purposes and shouldn't be used by the user.

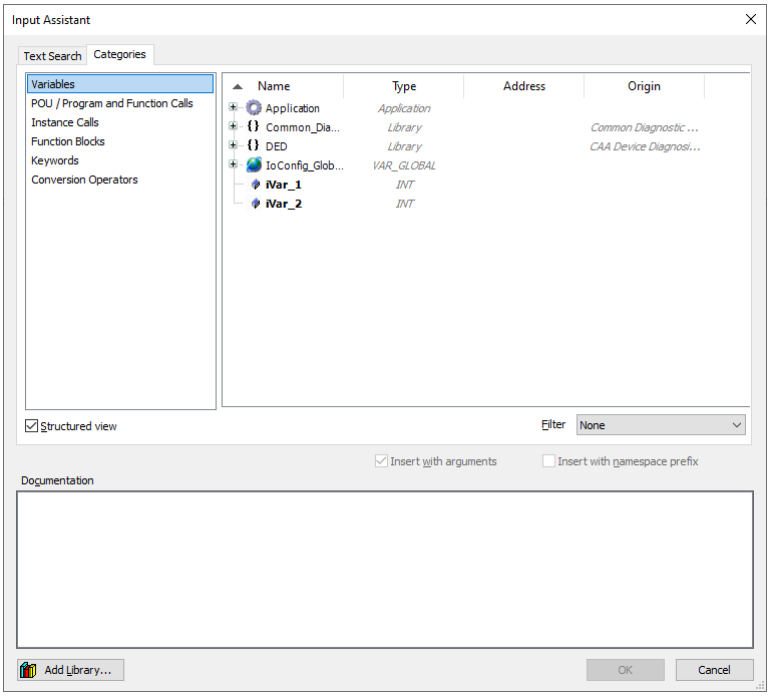

Figure 72: Categories Tab

The *Insert with Arguments* and *Insert with Namespace Prefix* options and the *Documentation* field are available in the *Text Search* and *Categories* tabs.

If option *Insert with arguments* is activated, items which include arguments, like for example functions, will be inserted with those arguments. For example: If function block FB1, which contains an input variable fb1_in and an output variable fb1_out, is inserted with arguments, the following will be written to the editor: fb1(fb1_in:= , fb1_out=>).

With the option Insert with context prefix enabled, the item will be inserted with the prefixed namespace. Currently, this option is available only for global variables.

If the selected element is a variable with an assigned address and there's a comment added to this declaration, this items are going to be displayed in the *Documentation* field.

7.2.10. Auto Declare...

Default Shortcut: <SHIFT>+<F2>

This command opens the *Auto Declare* dialog for the declaration of a variable. For this purpose, the cursor must be placed in a line of the implementation part of the editor, which contains an undeclared variable or an already variable must be selected. If the dialog should open automatically as soon as a line containing a not yet declared variable is left, the respective option in the SmartCoding must be activated.

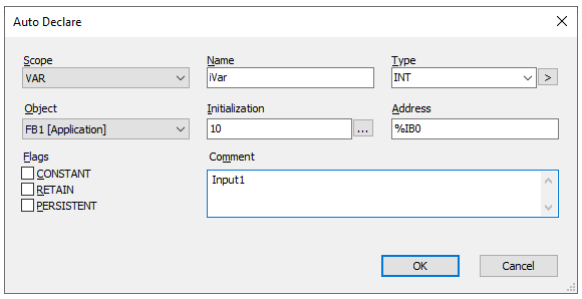

Figure 73: Dialog Box for Declaration of Variables

Some fields will be filled automatically with default values, but still can be edited. See below:

- *Name*: By default, the name of the new variable which you have entered in the editor.

- *Object*: : By default, the name of the currently edited object. To define another object where the variable declaration should be performed, select one of the available objects. For example, if you are going to declare a global variable (Scope: VAR_GLOBAL), here you will get all global variables lists already defined within the project.
- *Type*: By default, INT. If this is the first variable in the line: INT, but, if there is already a declared variable in the line, the type of this variable will be pre-set.

For modifying this entry you can press button to get the *Input Assistant* dialog, which allows you to select one from all possible data types. In case you want to declare an ARRAY variable you might use the array wizard, which is offered also via the arrow button. See below for a description.

- *Scope*: By default, VAR (local variable). Alternatively set another scope from the selection list.
- *Initialization*: Here you can enter an explicit initialization value for the variable. If nothing is entered here, the variable will be initialized with the default value.
- *Address*: The variable being declared can be bound to an IEC address (*AT* declaration). Example: variable *iVar* of type *INT* and address *%IB0* -> declaration: *iVar AT %IB0 : INT;*.
- *Comment*: If applicable, enter a comment. The comment text can be formatted with line breaks by using the key combination <CTRL>+<ENTER>. It will appear in the declaration part of the object in the line above variable declaration.
- *Flags*(CONSTANT, RETAIN, PERSISTENT): Activate the desired option to define whether the variable is a constant or a remanent variable. The appropriate attribute is added to the VAR keyword, e.g. *VAR CONSTANT*, which starts the declaration part for the variable. The PERSISTENT option is only available if a list of persistent variables exists.

7.2.10.1. Autodeclaration of Arrays

If you want to use the wizard for the declaration of ARRAY variables, use the arrow button ( ) behind the Type field and select command Array Wizard. The Array dialog will open:

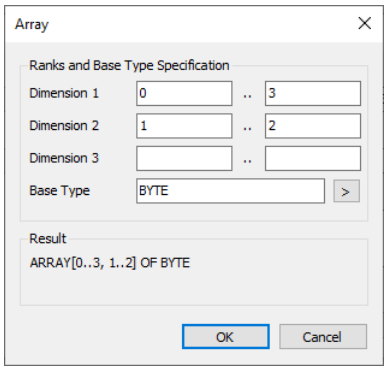

Figure 74: Array

At least the fields marked with an exclamation mark ( ) must be filled. Define the dimensions by entering the lower and upper limits and the base type of the variable, whereby the arrow button can be used to get to the input assistant.

In the lower part of the dialog (Result), a preview of the currently configured array declaration will be viewed.

See also the help page on **ARRAYS** (IEC 61131 Programming Manual).

By pressing *OK*, the declaration dialog will be closed and the variable declaration will appear in the declaration editor in accordance to the IEC syntax

7.2.11. Messages View

The *Messages View* commands allow the navigation between messages displayed in the messages window and also between messages and the implementation code.

Available commands:

- Next Message (F4)
- Previous Message (<SHIFT> + F4)
- Go To Source Position

These commands serve to navigate between messages in the *Messages* window (messages view) and also between messages and the concerned position in the project.

7.2.12. Refactoring

Menu and sub-menus *Refactoring* are displayed in *Edit* menu depending on which variables (or set of variables) are being selected. Variables can be renamed, added, removed, reordered and even some languages can have their pins of their blocks updated.

The commands are:

- Rename '<Var>'..
- Add Variable..
- Remove '<Var>'..
- Reorder Variables..
- Update Referenced Pins

Where <*Var*> is the name of the variable.

7.2.12.1. Rename '<Var>'...

Symbol: 

When you select an object (either in the device tree or in any POU) and execute the command, a window called *Rename* will open. Here you can see the object's old name and choose a new one. To finalise the process, just click *OK*

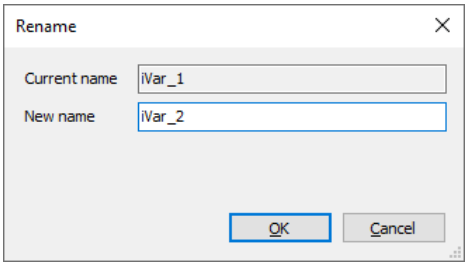

Figure 75: Rename

7.2.12.2. Add Variable...

Symbol: 

The command allows you to add a variable to a POU. To do this, simply select the declaration part and execute the command. The same window as *Auto Declare* will open, for more information check out Auto Declare..

7.2.12.3. Remove '<Var>'...

Symbol: 

The cursor need to be positioned on the variable that will be removed in the declaration part. Once the command is executed, the *Remove Variable* window opens showing the name, type and where the variable was defined. The user is also asked if they would like to complete the action by removing the variable and updating all references. Clicking *OK* will do this, clicking *Cancel* will cancel the operation.

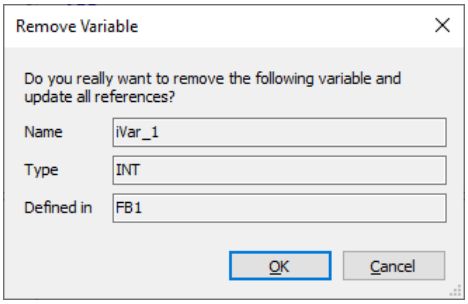

Figure 76: Remove Variables

7.2.12.4. Reorder Variables...

Symbol: 

The command allows to change the order of the variables in the declaration editor for the selected scope: VAR_INPUT, VAR_OUTPUT, or VAR_IN_OUT. To do this, simply select the statement and execute the command. This will open the *Reorder* window. Here you can readjust the order of the variables and select the scope.

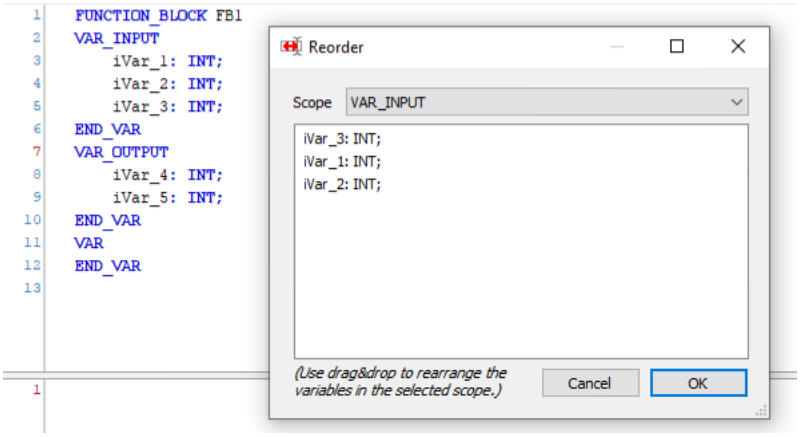

Figure 77: Reorder Variables

7.2.12.5. Update Referenced Pins

The described command updates the pins of a function block to align with the latest block declaration. This adaptation occurs in all relevant locations where the block is used. To execute this command, navigate to the *Edit* menu and select the *Refactoring* option, or use the context menu. It is essential that the cursor is positioned within the name of the function block either in the first line of the block declaration or within the device tree for the command to be available.

ATTENTION

The command is only enabled in CFC, FBD, LD, and IL. It is a combination of *Reset connections* and *Update parameters*.

7.3. View Menu

- Devices
- POUS
- Messages
- Element Properties
- Product Library
- I/O Simulator
- Toolbox
- Watch
- Cross Reference List
- Call Tree
- Bookmarks
- Breakpoints
- Call Stack
- Start Page
- Security Screen
- Full Screen
- Properties...
- Visualization ToolBox

7.3.1. Devices

Symbol: 

Default Shortcut: <ALT>+<0>

In the Devices view window all devices needed for the project are configured and the applications are defined appropriately.

7.3.2. POUS

Symbol: 

Default Shortcut: <ALT>+<1>

In the POUs view window all programming units of the current project (PLC program) are organized and can be instantiated for the use in an specific application.

7.3.3. Messages

Symbol: 

Default Shortcut: <ALT>+<2>

This command opens the Messages window.

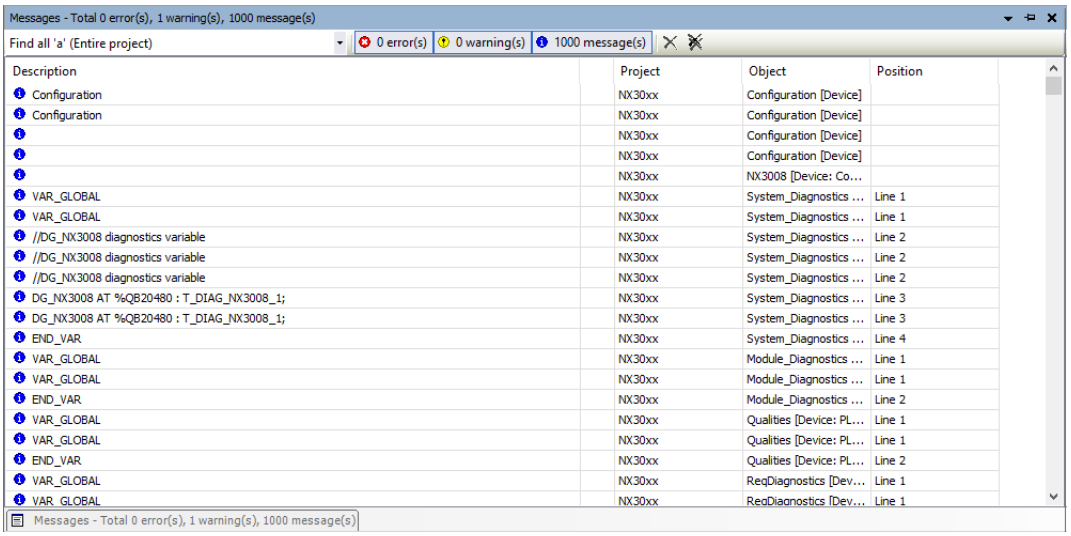

Figure 78: Search Results for String *a*

Messages might describe errors (), warnings () or just information ().

Further on messages are categorized after the concerned component or functionality. For example messages on syntactical checks of the project are generated in categories *Precompile*, messages on the compilation of the project in category *Build* (for example compile errors, code size).

There also might be messages on the import of a project, on the Library Manager etc.

You can select the desired message category in the selection list below *Messages* (figure above).

The messages belonging to the chosen category will be listed in the Messages table with the following information: *Description* (message text), *Project* (project name), *Object* (name of concerned object within the project), *Position* (for example line number, network number etc. within the object).

If you want to fade out or in a certain type of messages in the table, use the buttons in the upper right corner: *error(s)*, *warning(s)*, *message(s)*. These buttons in each case show the number of available messages and by a mouse-click on a button you can toggle the display of the respective message type.

You can navigate between the messages currently shown in the table and jump from a message to the position in the concerned object by using the commands *Next Message*, *Previous Message* and *Go To Source Position* (see Messages View for further details).

7.3.4. Element Properties

Symbol: 

This command opens the *Properties* view for the currently selected SFC element. The properties, like step or transition name, comment, step time attributes and associated actions are displayed in a structured table. They can be edited by a mouse-click in the values field and in case of the *Init Step* property by a click on the checkbox to activate or deactivate the option.

See **SFC Element Properties** (IEC 61131 Programming Manual) for details on the particular element properties.

7.3.5. Product Library

Symbol: 

This command opens the *Product Library* view, where the user can choose and insert devices in the project, for further information see Adding Modules.

7.3.6. I/O Simulator

Symbol: 

The command simulates module inputs and outputs.

For the command to be enabled, you must be logged into the PLC, as you can see in the figure below, with an NX3008, an NJ1001 (Digital Input) and an NJ6000 (Analog Input).

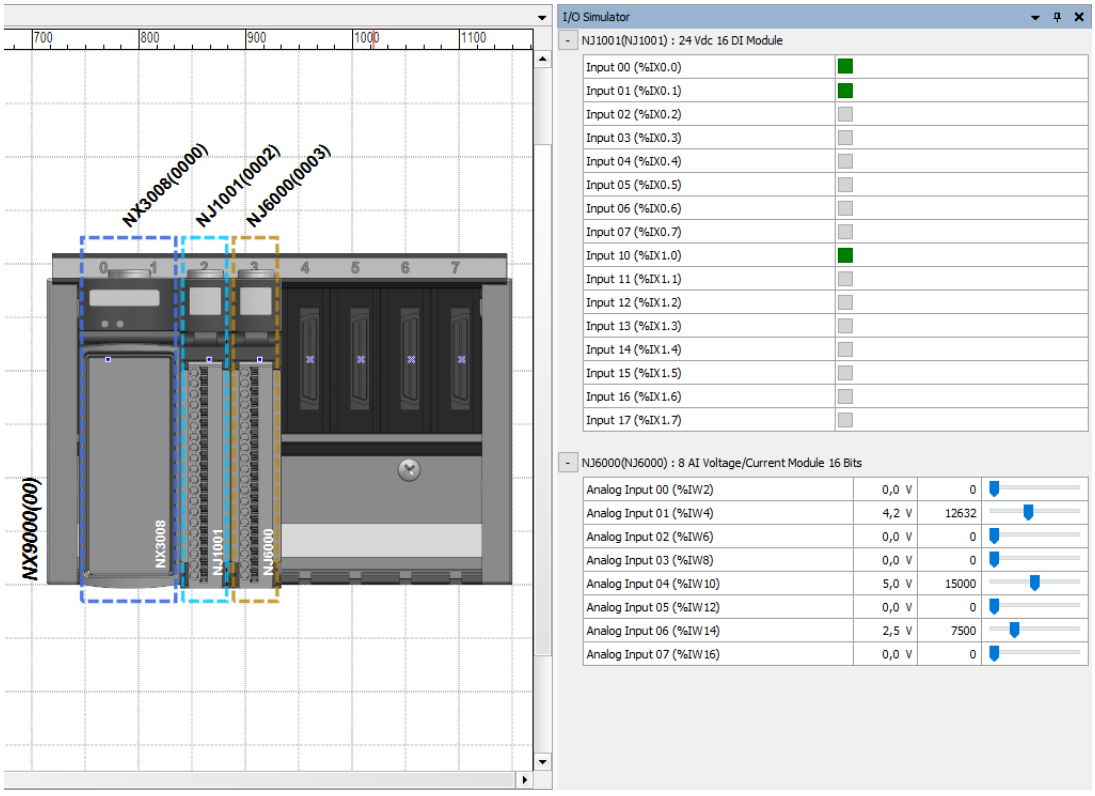

Figure 79: I/O Simulator

In this way, you can stimulate the variables by keeping them in the correct value window

Also, with output modules, it's possible to see the values set in the variables.

7.3.7. Toolbox

Symbol: 

This command opens the toolbox for the currently used editor in a window. Typically, toolboxes are available with graphic language editors or the visualization editor and provide graphic programming elements, which can be inserted into the editor via drag & drop.

7.3.8. Watch

Symbol: 

This command opens a submenu with commands *Watch 1*, *Watch 2*, *Watch 3*, *Watch 4* and *Watch all Forces*. These are used to open the respective watch list in a view window. *Watch all Forces* is a special watch view for the currently forced values. For further information, see Debug Menu.

7.3.9. Cross Reference List

Symbol: 

The command opens the *Cross Reference List* view. See Display Cross References for more information.

7.3.10. Call Tree

Symbol: 

The command opens the *Call Tree* view. For more information see Display Call Tree

7.3.11. Bookmarks

Symbol: 

The command opens the *Bookmarks* view. The view includes a list of bookmarks with the following details: *Bookmark*, *Object*, and *Position*.

You can rearrange the order of the bookmarks by dragging and dropping them.

Double-clicking a line will open the corresponding Object in the editor and jump to the bookmark.

You can also jump to the next bookmark (), to the previous bookmark () and delete the bookmarks ().

If you jump to a bookmark and the editor is closed, MasterTool IEC XE opens the editor and redirects the cursor to the bookmark line.

7.3.12. Breakpoints

Symbol: 

This command opens the *Breakpoints* dialog, which provides an overview on all breakpoints currently set in the project. The current breakpoint parameters are displayed and can be modified. In addition, breakpoints can be added, removed, enabled or disabled.

The breakpoint parameters basically are those which have been set when creating a breakpoint by the *Toggle Breakpoint* command (default condition parameters are assigned) or via the *New Breakpoint* (possibility to define certain conditions). For further information, see Debug Menu.

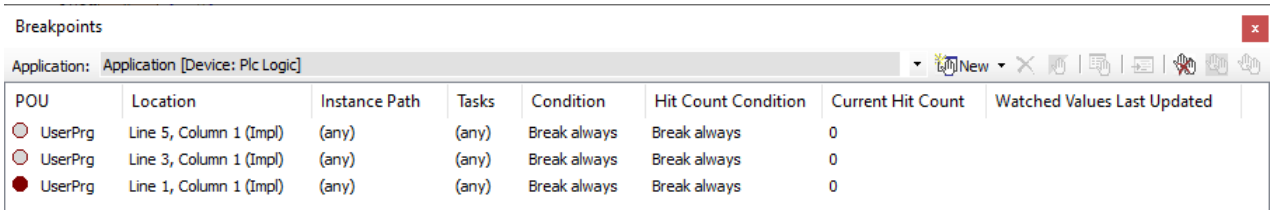

Figure 80: Breakpoints

- *Application*: Name of currently active application. Example: *Application [Device:PlcLogic]*.
- POU: Name of POU containing this breakpoint. Example: *UserPrg*.
- *Location*: Breakpoint position within POU: line+column numbers (text editors) or network or element numbers (graphic editors); *(Impl)* in case of function blocks indicates that the breakpoint is in the implementation part of the function block. Example: *Line 2, Column 1 (Impl)*.
- *Instance Path*: Complete object path of the breakpoint position. Example: *Device.Application.UserPrg*.
- Tasks: Tasks during the run of which the breakpoints should be noticed: *(n)* in case of no restriction (default) and particular task name(s). Example: *MainTask, SubTask1*.
- *Condition*: Definition of when (number of hits) the breakpoint should cause a break in processing; possible entries see New Breakpoint... Example: *Break when the hit count is equal to 3*.
- *Current Hit Count*: Indicates how often the breakpoint has been run through (hit) up to now. Example: *3*.

The following functions are available as buttons in the upper right part of the dialog for editing the current breakpoints parameters and for removing or adding breakpoints.

7.3.13. Call Stack

Symbol: 

This command opens the *Call Stack* window. When you are stepping through a program in online mode, always the currently reached step position will be indicated there with its complete call path. The Call Stack window below the title bar always displays the name of the currently concerned Application and the name of the Task controlling the currently reached POU.

The call stack is displayed as a list of positions, each described by POU name, Location and - in case of instances - with the *Instance Path*. Depending on the editor, the location is described by the line and column numbers (text editor) or by the network or element numbers (graphic editors).

The first line in this list, indicated with a yellow arrow, describes the current step position. If this position is within a POU, which is called by another POU, the position of the call will be described in the next line. If this POU again is called by another POU, the call position follows in the third line, and so on.

The call stack view is also available in offline mode and during normal online run. In this case The position which was last viewed during an online stepping session will be still displayed, but in greyed letters.

Figure 81: Call Stack View

7.3.14. Start Page

Symbol: 

This command opens a view providing a selection of commands for quick starting with a new or recent project, version information, and a viewer for the Altus home page.

In the Load and Save options you can configure, that the start page automatically appears when the programming system gets started.

7.3.15. Security Screen

Symbol: 

The command opens the *Security Screen* view. The MasterTool IEC XE provides several security features, which are configured and displayed in the interface. These features include personal user certificates, encrypted communication, and encryption and signatures for IEC projects. Additionally, the system supports the encryption and signature of downloads, online changes, and boot applications. The overall security level is also indicated within the view.

There are three tabs where this features can be configured, they are *Users*, *Project* and *Devices*.

- *Users*: On this tab, you can configure certificates necessary for encrypted communication and digital signatures of the user. Only certificates containing private keys can be specified here. The user profile is saved as an XML file in the user options.
  - *User Profile and Certificate Selection*: By default, the user profile is set to the Windows login name.
  - *List box with existing user profiles*:  Opens the *User Profiles* dialog, where you specify the name for a new user profile. To delete a profile, select it and use the button  .
  - *Digital Signature*: To open the *Certificate Selection* dialog for selecting the certificate for the digital signature, use the  button. To delete the displayed certificates, selected them and use the  button. Only one certificate can be selected. The certificate has to have a private key.
  - *Project File Decryption*: To open the *Certificate Selection* dialog for selecting the certificate for decrypting project files, use the  button. To delete the displayed certificates, selected them and use the button. Only one certificate can be selected. The certificate has to have a private key.
  - *Enforce encrypted communication*: If checked, when the user communicates with the PLC, the server certificate of the PLC is used to establish an encrypted connection, ensuring that all communication is encrypted.
  - *Enforce encryption of project files*: If checked, all project files are encrypted with a certificate. When the project is saved, it is encrypted using the certificate specified in the *Project Settings* under the *Security* dialog. The selected certificate is displayed in the *Project File Encryption* group on the *Project* tab. To open this project, the certificate with a private key must be specified in *Project File Decryption*.
  - *Enforce signing of project files*: If checked, all project files are signed with a certificate. In the Digital Signature settings, a certificate with a private key must be specified. When a project is saved, a signature file (<project name>.project.p7s) is generated in the project directory, containing the signature.
  - *Enforce encryption of downloads, online changes and boot applications*: The data downloaded to the PLC must be encrypted with a controller certificate. This certificate can be defined either in the properties dialog of the application under the *Encryption* tab or on the *Project* tab in the *Encryption of Boot Application*, *Download*, and *Online Change group*. PLC certificates are stored in the local Windows Certificate Store within the PLC Certificates directory. If the PLC certificates are not available in this directory, they must first be loaded from the controller and installed.
  - *Enforce signing of downloads, online changes and boot applications*: If checked, the online code (downloads, online changes, and boot applications) must be signed with a certificate that includes a personal key. The certificate is selected from the *Digital Signature* area.
  - *Enforce signing of compiled libraries*: If checked, the *File > Save Project as Compiled Library* command generates a signed library (<library name>.compiled-library-v3).
  - *Enforce timestamping of signed compiled libraries*: The URL of the time stamp server that created the time stamp must be entered in the *Timestamping* server field. The CheckBox is only enabled when it's checked to sign compiled libraries.
- *Project*: All project-specific settings are configured on this tab and are only active when a primary project is loaded.
  - *Technology*: When you choose the *Encryption* project setting and then select *Certificates* in the dialog, you can click  to choose the appropriate certificate.
  - *Certificates of Users Sharing this Project*: Area for listing the certificates that encrypt the project file.
  - *Encryption of Boot Application, Download and Online Change*: Double-clicking an application in the list opens the *Properties > Security* dialog. The fields available in the open properties dialog depend on the settings of the Security Level on the *User* tab of the *Security Screen*. These fields may include an Encryption tab with an active Certificates area and an Encryption tab with an Encryption Technology list box. In the Properties > Encryption dialog, click the button represented by the  icon to select the controller certificate for the encryption of Boot Application, Download, and Online Change.
- *Device*: You can configure encrypted communication with the PLC and encrypt the boot application, downloads, and online changes.

7.3.16. Full Screen

Symbol: 

Default Shortcut: <CTRL>+<SHIFT>+<F12>

This option, if activated, effects that the MasterTool IEC XE frame window will be displayed in full-screen mode. To toggle back to the previous mode deactivate the menu entry or press the shortcut again.

7.3.17. Properties...

Symbol: 

This command opens dialog Properties <objectname>. The properties of the currently selected object in the POUs or Devices view will be displayed on various tabs, the availability of which depends on the type of object. The following dialogs are possible.

7.3.17.1. Common

It provides information on the object.

- *Full name*: Object name as used in the POUs or Devices View Object type.
- *Object type*: Type of the object, for example POU, Application, Interface etc.
- *Open with*: Type of the editor, which is used to edit the object.

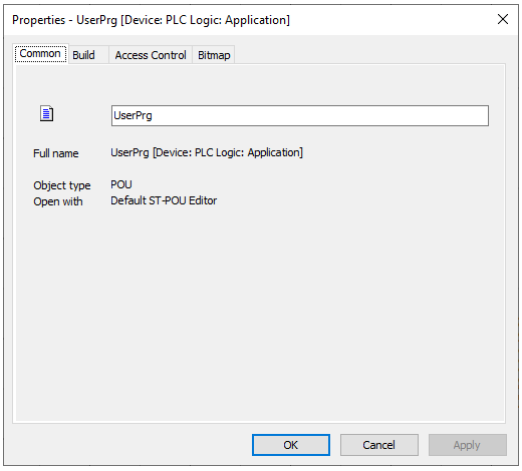

Figure 82: Properties Dialog, Common Category

7.3.17.2. Build

Concerning the compilation (build) the following options can be activate.

- *Exclude from build*: The object will not be noticed during the next *Generate Code* command.
- *External implementation (Late link in the runtime system)*: No code is created for this object during a compilation of the project. The object will be linked when running the project on a target, if it is available there, for example via a library.
- *Enable system call*: Background: In contrast to MasterTool IEC XE previous versions now the ADR-Operator can be used with function names, program names, function block names and method names, thus replacing the INSTANCE_OF operator. See in this context **Function** pointers (IEC 61131 Programming Manual). HOWEVER, there is no possibility to call a function pointer within MasterTool IEC XE . In order to enable a system call (runtime system) you must activate the current option for the function object.
- *Compiler defines*: Here you can enter *defines* (see define instruction) and conditions for the compilation of this object. The expression *expr* used in those pragmas can be entered here, several entries can be entered in a comma-separated list. Nowadays the *Compiler defines* is disable for the user.

For example, it might be useful to make dependent the compilation of an application on the value of a certain variable.

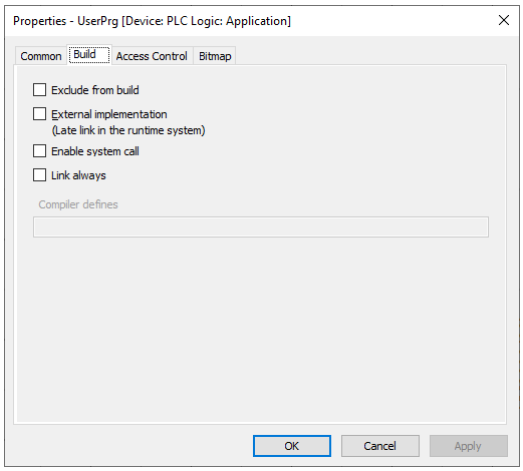

Figure 83: Properties Dialog Category Build

7.3.17.3. Access Control

This dialog allows configuring the access rights on the current object for the available user groups. This corresponds to the configuration via the *Properties...* dialog, which is available in the *Access Control* menu.

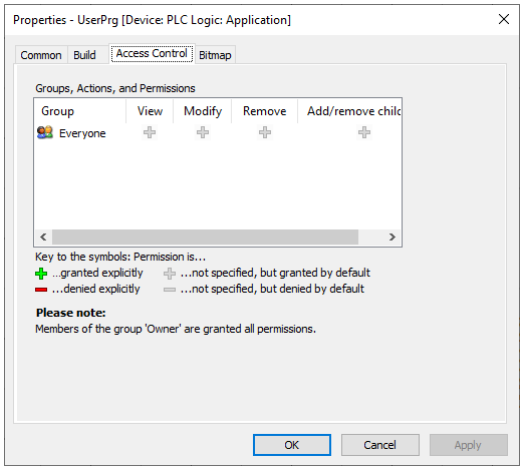

Figure 84: Properties Dialog, Category Access Control

To edit the right for a certain action and group select the respective field in the table, perform a mouse-click or use <spacebar> to open the selection list and from there choose the desired right.

For a description on possible actions, rights and the symbols please see the help page on the *Permissions* dialog.

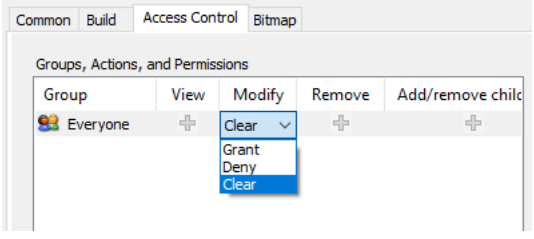

Figure 85: Selection List of Rights for action *Modify* for group *Everyone*

7.3.17.4. Boot Application

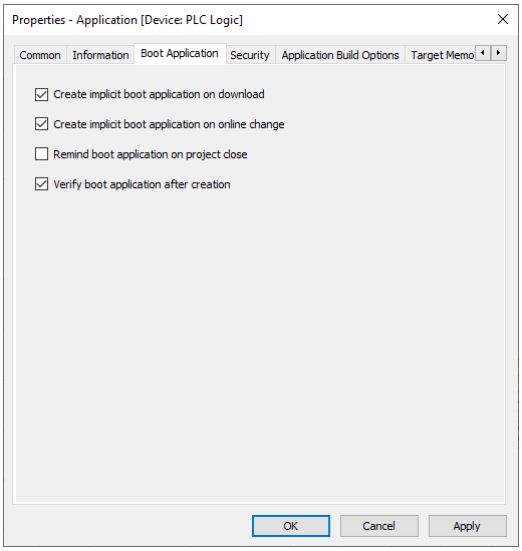

Figure 86: Properties, Boot Application Category

- *Create implicit boot application on download*: If activated, at a download of the project automatically a boot application will be created.
- *Create implicit boot application on online change*: If activated, at an online change automatically a boot application will be created.
- *Remind boot application on project close*: If activated, when going to close the project you will be asked whether the boot application should be updated/created.
- *Verify boot application after creation*: Once the boot application is created, a separate service verifies whether it was created correctly.

7.3.17.5. Link to File

Global variables lists can be defined with the help of an external file in text format. Such a file can be generated by using the export functionality provided in the *Properties* dialog of the respective variables list: If option *Export* before compile is activated, automatically at each project compilation, a file with extension *.gvl* will be created and be stored at the path specified in the *Filename* field. If option *Import* before compile is activated, an existing list export file can be read at each project compilation. This allows to import a GVL created from another project, for example in order to set up network variables communication.

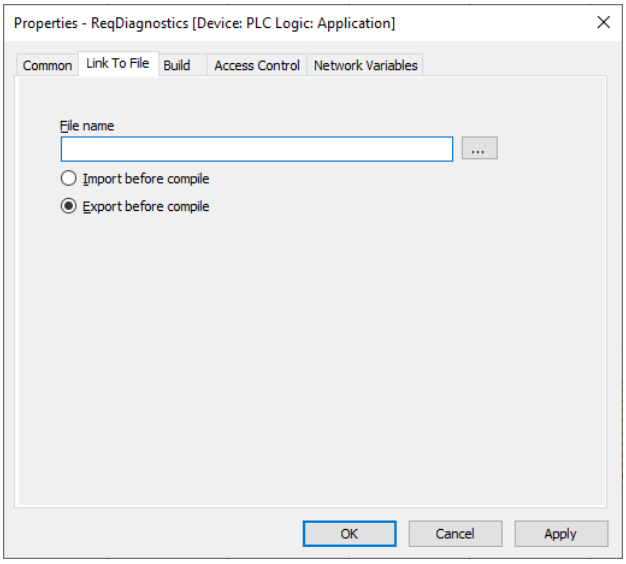

Figure 87: Properties Dialog, Category Link to File

7.3.17.6. SFC Settings

This dialog allows settings for the current SFC object concerning compilation and flag handling. The items handled in the tabs *Flags* and *Build* correspond to those handled in the SFC options dialog, where the default settings for SFC objects are defined. See the related item for a description of the particular settings.

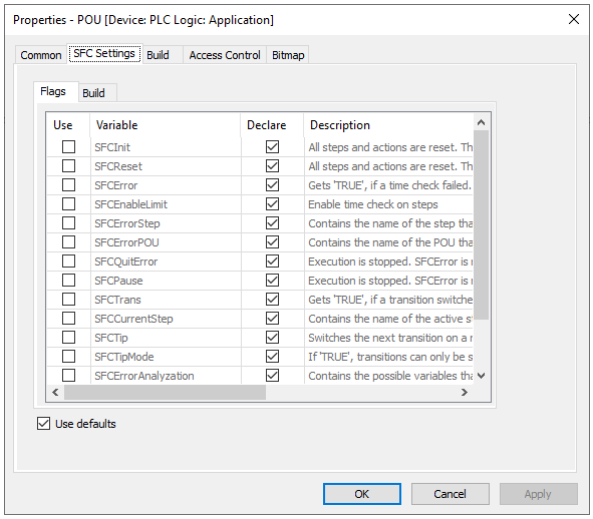

Figure 88: Properties Dialog, Category SFC Settings, Flags

The *Set Defaults* button will exactly apply those defaults, which are currently defined in the *SFC Options* dialog, to the current object.

7.3.17.7. External File

This tab of the Properties dialog for external files added to the project lets you view and modify the properties that were set in the dialog box *Add External File*. For further information see External File.

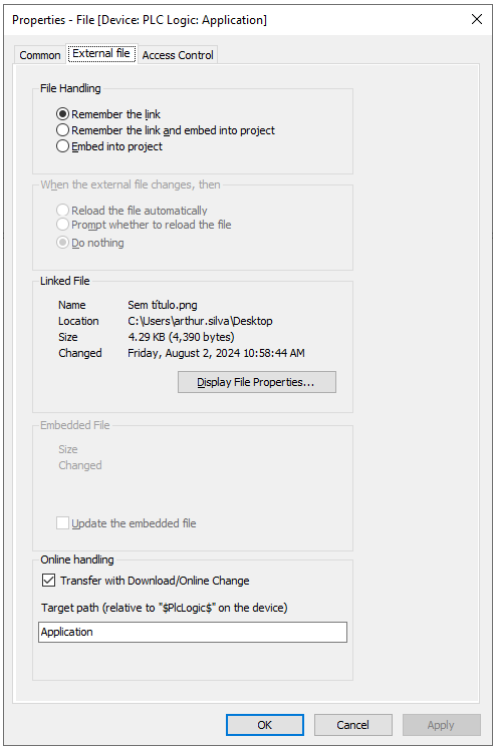

Figure 89: PropertiesDialog, ExternalFile

7.3.18. Visualization ToolBox

Symbol: 

The view provides the visualization elements which you can insert into the active visualization editor.

7.4. Project Menu

Provides commands to manage the objects and folders of the project.

Available commands:

- Add Object
- Add Folder
- Scan for Devices..
- Edit Object
- Edit Object with
- Edit Object (Offline)
- Project Information
- Project Settings
- Project Update
- Document
- Compare
- Commit Accepted Changes
- Compare
- Export PLCopenXML
- Import PLCopenXML
- Import Safety Project..
- Delete Safety Objects Imported..
- Restore Points
- User Management
  - User Login..
  - User Logout
  - Permissions..

7.4.1. Add Object

Symbol: 

This command opens a submenu providing the objects available for getting inserted at the currently selected position in the *POUs* or *Devices* tree.

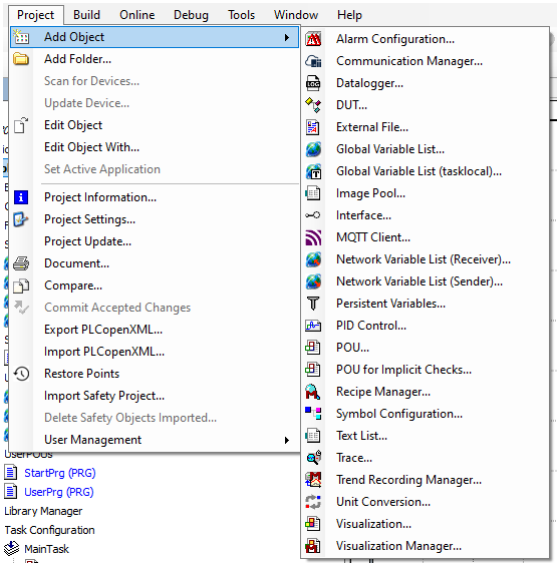

Figure 90: *Add Object* Submenu

Select the desired object type and in the appearing dialog define a Name. Please notice the given recommendations on the naming in order to get the name as unique as possible. Depending on the object type also further configuration settings might be available. For further information on this please see the respective the particular object type and the corresponding editor.

This command is also available in the context menu of the *POUs* tree and *Device* tree.

To rename an object in the POUs tree, click on the entry to open an edit frame or use the *Properties* dialog.

7.4.1.1. Alarm Configuration...

Symbol: 

The *Alarm Configuration* is the object responsible for configuring the alarms. When the object is inserted, the following objects are inserted automatically:

- Alarm class: Error
- Alarm class: Info
- Alarm class: Warning
- Alarm storage: AlarmStorage

You have the option to use these objects, but it's not required. You can delete them and replace them with objects of your own choosing.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Visualization/_cds_obj_alarm_configuration.html.

7.4.1.2. Communication Manager...

Symbol: 

The communication path objects of OPC UA are inserted below the *Communication Manager* object.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Communication/_comm_communication_manager.html.

7.4.1.3. Datalogger

Symbol: 

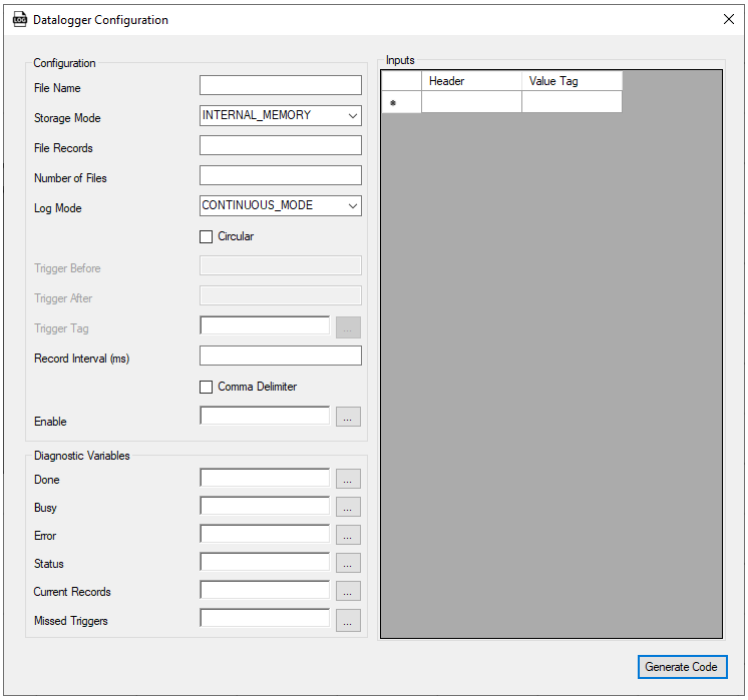

Figure 91: Datalogger Configuration

Dialog to configure each *Datalogger* setting. For more detailed information about configuring and using the Datalogger, please refer to the specific manual for the LibDatalogger.

The object has a *Configuration* window and a *Inputs* window. These are the *Configuration* settings:

- *File name*: Name of the CSV file that will be generated.
- *Storage Mode*: Can enable storage of data in three different ways: Internal memory, memory card and usb mass storage.
- *File Records*: Number of data records in data log.
- *Number of Files*: Maximum number of files created by Datalogger.
- *Log Mode*: Set the log mode between *CONTINUOUS_MODE* and *TRIGGER_MODE*.
- *Circular*: Circular logger.
- *Trigger Before*: Enabled if the log mode is *TRIGGER_MODE*. Number of logging lines before the trigger.
- *Trigger After*: Enabled if the log mode is *TRIGGER_MODE*. Number of logging lines after the trigger.
- *Trigger Tag*: Occurrence of trigger.
- *Record Interval (ms)*: Interval in which the data will be recorded.

- *Comma Delimiter*: If true, the CSV columns are delimited by commas and decimal point by a dot; otherwise the CSV columns are delimited by semicolons and decimal point by a comma.
- *Enable*: Enable function block.
- *Done*: If it's TRUE, Datalogger finished its operation.
- *Busy*: If it's TRUE, Datalogger is capturing the inputs and writing the log files.
- *Error*: If it's TRUE, indicates that an error has occurred with the function block.
- *Status*: Output that identifies the internal status.
- *Current Records*: Current number of data records.
- *Missed Triggers*: Trigger Counter ignored by Datalogger.
- *Generate Code*: Create a POU named *Datalogger* with the configurations setted on the dialog.

The *Inputs* window have a *Header* column and a *Value Tag* column. The *Header* is the header of the data, while *Value Tag* is the linked variable that will be the input data.

7.4.1.4. DUT

Symbol: 

User defined data types can be created in the *Data Unit Type* editor (DUT editor). This is a text editor and behaves according to the currently set text editor options.

The DUT editor will be opened automatically in a window when adding a DUT object in the *Add object* dialog. In this case it provides by default the syntax of an extended structure declaration, which then might be changed as desired to a simple structure declaration or to the declaration of another data type unit, for example an enumeration.

The editor also opens when you open an existing DUT object currently selected in the *POUs* view.

> **ATTENTION**
>
> For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_obj_dut.html.

7.4.1.5. External File...

An external file refers to any file that is added to the project through the POUs or Devices view. To add an external file, go to *Project* > *Add Object*, which opens the *Add External File* dialog. This dialog allows you to specify how the file is associated with the project.

The *Transfer with Download/Online Change* option enables the configuration of external file transfers.

Please note that when an external file is downloaded to the PLC, it will not be updated within the project.

> **ATTENTION**
>
> For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_obj_external_file.html

7.4.1.6. Global Variable List...

Symbol: 

The GVL Editor is a declaration editor for editing Global Variables Lists. It is working according to the options currently set for a text editor and in online mode also appearing like described for the declaration editor. The declaration must start with *VAR_GLOBAL* and end with *END_VAR*.

These keywords are provided automatically. In between enter valid declarations of global variables.

> **ATTENTION**
>
> For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_obj_gvl.html.

7.4.1.7. Global Variable List (tasklocal)...

Symbol: 

A *Global Variable List (tasklocal)* allows for the declaration and editing of global variables that can be written by only one task, while all other tasks have read-only access. This ensures consistent variable values, even in multicore projects.

> **ATTENTION**
>
> For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_obj_gvl_tasklocal.html

7.4.1.8. Image Pool...

Symbol: 

An image pool is a table that lists image files with their ID, file name, thumbnail, and link type. You can add and edit images centrally in this pool. In your code, you reference images by their ID instead of the file name. This central storage is mainly used for visualizations.

> **ATTENTION**
>
> For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Visualization%20Support/_vsprt_using_image_pool.html

7.4.1.9. Interface

Symbol: 

An interface in object-oriented programming defines a set of method and property prototypes, meaning it includes only declarations without implementations. This allows function blocks with common properties to be used interchangeably. To add an **ITF** object to your application or project, select *Project* > *Add Object* > *Interface*.

> **ATTENTION**
>
> For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_obj_interface.html

7.4.1.10. MQTT Client

Symbol: 

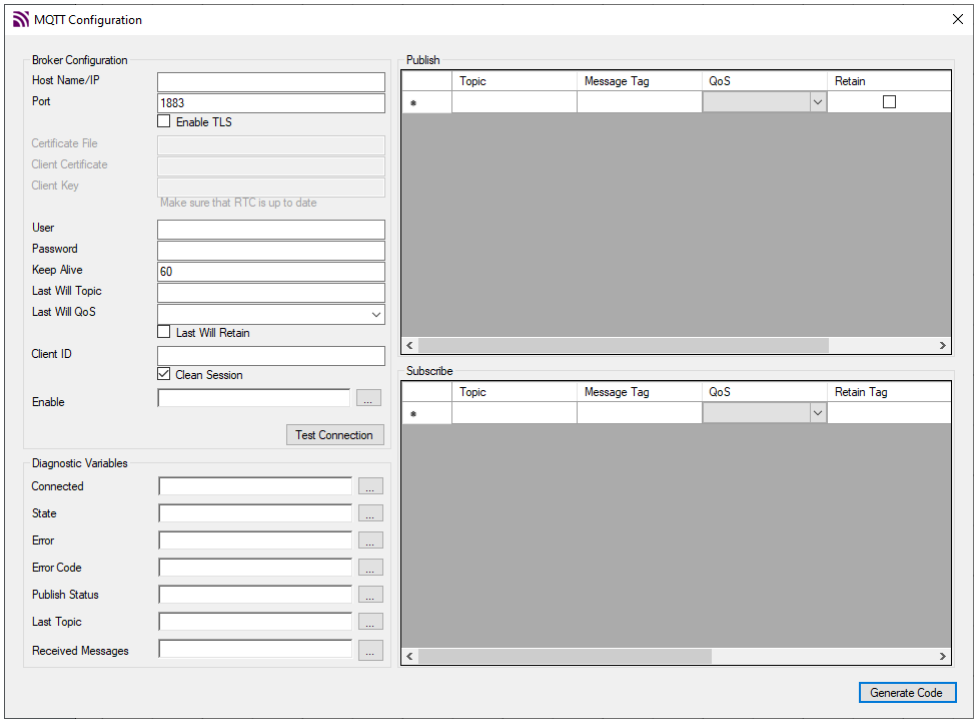

Figure 92: MQTT Configuration

This object creates a POU named MQTTClient, which communicates the selected variables with the configured host using the MQTT protocol. The dialog automatically applies the specified settings. For more detailed information about configuring and using the MQTT protocol, please refer to the specific manual for the LibMQTT.

- *Host Name/IP*: MQTT broker address. Can be an IP or URL.
- *Port*: Broker port (default: 1883; use 8889 for TLS).
- *Enable TLS*: Enables TLS (Transport Layer Security). TLS is a security protocol.
- *Certificate File*: Specifies the Certificate Authority file name (must be exported to the cert folder in Devices/Files).
- *Client Certificate*: Specifies the client certificate file name (mandatory if required by the broker).
- *Client Key*: Specifies the client key file name (mandatory if required by the broker).
- *Username*: Optional broker username.
- *Password*: Optional broker password.
- *Keep Alive*: Keep Alive is a mechanism used in internet communication protocols to maintain an active connection between a client and a server. It ensures the connection remains open, allowing multiple requests and responses to be exchanged without repeatedly establishing new connections. Here, you can configure the duration (in milliseconds) for which the connection should remain active.
- *Last Will Topic*: Defines the topic name for the "Last Will" message, sent if the client disconnects unexpectedly.
- *Last Will QoS*: MQTT Quality of Service (QoS) defines the level of guarantee for message delivery between sender and receiver. There are three QoS levels available:
  - *MQTT_QOS_0*: Also known as QoS 0, the message is delivered at most once, without requiring acknowledgment from the receiver. This level provides no delivery guarantee, making it the fastest but least reliable option.
  - *MQTT_QOS_1*: In QoS 1, the message is delivered at least once. The sender expects an acknowledgment from the receiver and will retry sending the message until the acknowledgment is received. This may result in the message being delivered multiple times. item *MQTT_QOS_2*: Known as the highest level of QoS, the message is delivered exactly once, ensuring no duplicates are received. This level uses a more complex handshake process between sender and receiver, making it the most reliable but also the slowest option.
- *Last Will Retain*: Optionally retains the "Last Will" message.

- *Client ID*: Unique identifier for the client (randomly generated if left empty).
- *Clean Session*: If TRUE, clears all subscriptions and messages on disconnect.
- *Enable*: Enables the function block.
- *Test Connection*: Verifies the connectivity between the PC and the broker using the specified host and port. Note: This test is performed directly by the PC, not the PLC, and is incompatible with TLS-enabled connections.
- *Connected*: Indicates a successful connection.
- *State*: Displays the function block's status.
- *Error*: Indicates an error.
- *Error Code*: Displays the error state, referencing the *MQTT_STATE* enum
- *Publish Status*: Displays the publishing status array.
- *Last Topic*: Shows the last topic received from the broker.
- *Received Messages*: Displays parameters of received messages.
- *Generate Code*: Creates a POU based on the dialog settings.

The tables on the right allow you to configure publish and subscribe variables. The *Publish* table includes the following columns:

- *Topic*: Topic name where the message will be published.
- *Message Tag*: Message to be published by the MQTT Publisher.
- *QoS*: MQTT Quality of Service, as already explained.
- *Retain*: If TRUE, retain the published message.

The *Subscribe* table, are the following columns:

- *Topic*: Topic name to be subscribed.
- *Message Tag*: Store the message received by the MQTT Subscriber.
- *QoS*: MQTT Quality of Service, as already explained.
- *Retain*: Received message retained.

7.4.1.11. Network Variable List (Receiver)...

Symbol: 

The object displays received network variables, showing details about the network, transmission, and sender information.

To include this object in an application, select *Add Object* > *Network Variable List (Receiver)*.

The *Network Variable List (Receiver)* shows the received network variables that were declared in the *Network Variable List (Sender)* of another device or project. Note that you cannot modify these network variables within the object editor.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_f_networkvariables.html

7.4.1.12. Network Variable List (Sender)...

Symbol: 

A *Network Variable List (Sender)* is used to declare and list global variables that will be sent to a *Network Variable List (Receiver)* on another device or network project.

To add this object to the device tree, click *Add Object* > *Network Variable List (Sender)* in the application.

You can configure the protocol and transfer parameters in the *Add Network Variable List (Sender)* dialog or in the *Properties* dialog of the object, under the *Network Variables* tab.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_obj_nvl_send.html

7.4.1.13. Persistent Variables...

Symbol: 

The object includes the declaration of global persistent variables within the **VAR_GLOBAL PERSISTENT RETAIN ... END_VAR** section. These variables are stored in a special non-volatile memory.

The persistence editor displays the variables in a standard list format. This displayed list does not affect the persistence behavior of the variables; it only reflects the list stored internally in the process image. This internal list contains all variables ever declared in chronological order. Variables that have been removed are marked with a placeholder and remain as gaps in the list.

The declaration section can also include instance paths that refer to locally declared persistent variables, which are created using the *Declarations* > *Add All Instance Paths* command.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_obj_gvl_persistent.html

7.4.1.14. PID Control

Symbol: 

This object creates a POU named *PIDControl*. When creating the POU, it's possible to edit with the *Default ST-POU Editor* or the *PIDControl*, that one opening a dialog to configure the settings that will be setted in the POU. If you choose to create by the *PIDControl*, it will also be possible to see a chart of the object in the dialog *Settings & Chart*.

In the *Advanced Configurations* tab, it's possible to set the SP, PV and MV variable names, as well as other configurations, such as *Sampling Time (ms)*.

7.4.1.15. POU...

Symbol: 

An object of type POU, or *programming organization unit,* is a crucial component in a MasterTool IEC XE project, where the source code for the controller program is written. The available POU types include Program, Function, and Function Block. To add a POU, you can navigate to the device tree or the POUs view and use the *Project* > *Add Object* command, specifying both the POU type and the implementation language. Additionally, you can incorporate other programming objects, such as methods and actions, within these objects.

When it comes to calling POUs, certain POUs have the capability to call others, though recursion is not permitted. MasterTool IEC XE follows a specific order when scanning the project for the POU to be called: it first checks the current application, then the Library Manager of the current application, followed by the POUs view, and finally the Library Manager in the POUs view.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_f_obj_pou.html

7.4.1.16. POU for Implicit Checks

Symbol: 

You can add these special POUs to an application to provide them with implicit monitoring functions. At runtime, these functions check the limits of arrays or subrange types, the validity of pointer addresses, and division by zero.

Note: This option can be disabled for devices that are already equipped with these kinds of monitoring blocks by a special implicit library.

The *Add Object* > *POU for Implicit Checks* command is used to add it to the application. The command opens the Add POU for Implicit Checks dialog where you can select a monitoring function type. Depending on the monitoring function, you have to edit the implementation code or create it yourself from scratch.

To prevent multiple inclusions, monitoring functions that have already been inserted are disabled in the Add POU for Implicit Checks dialog.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_f_obj_pous_implicit_check.html

7.4.1.17. Recipe Manager

Symbol: 

The *Recipe Manager* manipulates lists of user defined project variables named *Recipe Definition* and also value sets defined to these variables within a *Recipe Definition* named *Recipes*.

Value sets defined for variables of some *Recipes* may be interchanged (read or written in the PLC or loaded from other Recipes saved in files) in different moments for each situation. This way, they can be used for PLC control, industrial process control, variable backup, among other possibilities.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Recipes/_rec_user_interface.html.

7.4.1.18. Symbol Configuration...

Symbol: 

You can create symbol descriptions for project variables using the symbol configuration. To do this, go to *Project* > *Add Object* and add a symbol configuration object to the device tree. Afterward, you can define the specific default settings. Refer to the *Add Symbol Configuration* dialog below for more details.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Communication/_cds_obj_symbolconfiguration.html

7.4.1.19. Text List

Symbol: 

This object allows you to create, manage, and translate texts. It includes a table where you can add new entries. Texts composed here can be selected in a visualization under the Dynamic Texts property of an element. During runtime, the visualization dynamically displays the text in the chosen language.

When this object is assigned to an alarm group and placed under the *Alarm Configuration* object, MasterTool IEC XE automatically adds the alarm group's texts to the table. You can also manually add additional texts.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Visualization%20Support/_cds_obj_text_list.html

7.4.1.20. Trace

Symbol: 

The Trace functionality allows recording and reading the progression of the values of variables on the PLC over a certain time. For this purpose, the values of defined trace variables are steadily written to a MasterTool IEC XE buffer of a specified size and then can be viewed in the form of a graph along a time axis.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Trace/_trace_start_page.html.

7.4.1.21. Trend Recording Manager...

Symbol: 

The *Trend Recording Manager* object enables long-term data storage in a database during runtime, using the CmpTraceMgr runtime system component. In the device tree, this object serves as a node for trend recordings created under an application. Only one instance of this object can exist per application.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Visualization/_cds_obj_trend_recording_manager.html

7.4.1.22. Visualization

Symbol: 

This object represents a single visualization. You can insert it either below an application or make it available project-wide under the root node in the Devices view or directly in the POUs view. To edit the visualization, double-click the object entry in the device tree or the POUs view to open the visualization editor.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://www.helpme-codesys.com/codesys-visualization.html.

7.4.1.23. Visualization Manager

Symbol: 

The *Visualization Manager* handles the configuration settings for all display variants of the visualizations within the current application.

This object is automatically added below the application when a visualization object is inserted. Double-clicking it opens a configuration dialog with multiple tabs, with the *Settings* tab displayed by default.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Visualization/_visu_obj_manager.html

7.4.2. Add Folder

Symbol: 

This command will be available in the *Project* menu and also in the context menu if you are working in the POUs or devices view.

If you want to add a folder on the uppermost level in the objects tree of a view window, make sure that the window is active, but that no existing object or folder is selected. If you want to add a folder on a lower level, select that entry in the tree below which the folder should be inserted.

Then perform the command to open the *Add Folder* dialog and there define the name of the new folder. The name may contain spaces, digits and special characters.

7.4.3. Scan for Devices...

MasterTool IEC XE can perform searches for new devices in a field network by clicking over the master device of the field bus in the project tree and selecting the command *Scan Devices*.

Through this resource, it's possible to identify slave devices not yet configured in field networks such as PROFIBUS-DP, EtherCAT and CAN. Some specific conditions must be met to start the process in each network master. For further details, consult the User Manual of each master.

Once the search conditions are met, by executing the mentioned command the screen in the figure below will be displayed, showing the detected devices.

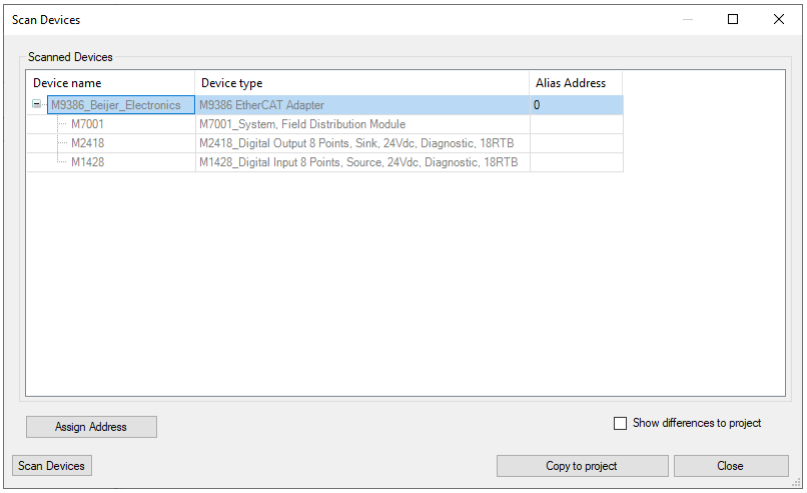

Figure 93: Scan for Devices Dialog

7.4.3.1. Show Differences for the Project

A part of the dialog window is visible only when you select the *Show differences to project* option. The differences between the scanned and configured devices are color-coded. Devices displayed in green are identical on both sides. Devices displayed in red are available only in the view of the scanned or configured devices. The referenced window can be seen below.

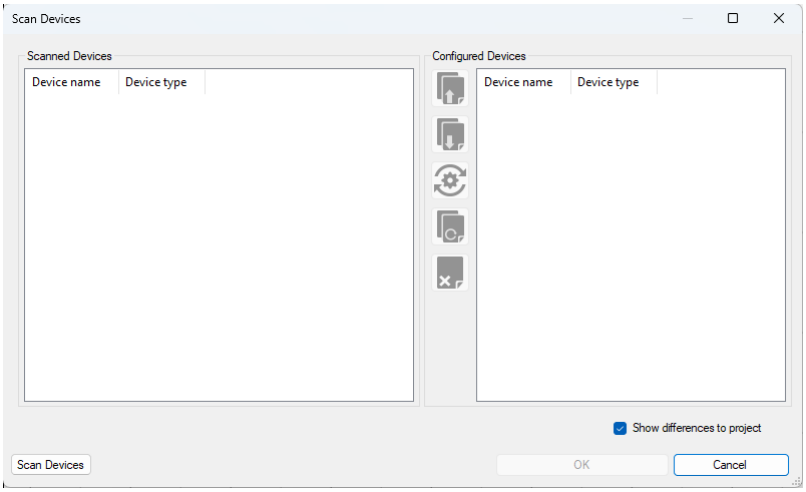

Figure 94: Show Differences for the Project

The window has some options for device manipulation, which are:

- Copy before: If you have selected a device in both views, then the scanned devices are inserted above the selected configured device.
- Copy after: If you have selected a device in both views, then the scanned devices are inserted below the selected configured device.
- Change to: If you have selected a device in both views, then the configured devices are replaced by the selected scanned device.
- Copy all: All scanned devices are copied to the project.
- Delete: Deletes the selected configured device.

If the detected device is present in the MasterTool IEC XE device repository it's possible to select it and, by pressing the *Copy to Project* button, a list of devices that can be added to the project will be shown. The selected objects will be inserted in the project in a similar way to what happen with the *Add Device* command.

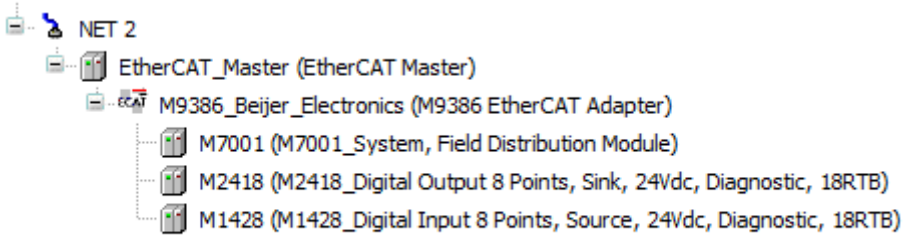

Figure 95: Devices copied with Scan for Devices

7.4.4. Update Device...

The *Update Device* command is used to change the version of a device that already exists in the device tree.

Update Device is only enabled for the following devices:

- Ethernet IP Scanner Adapter
- CAN open Remote

7.4.5. Edit Object

Symbol: 

Use this command if you want to edit or view an object, which is available in the POUs or Devices view. The object must be selected, then the command is available in the context menu and by default in the project menu.

The command will open the object in the appropriate editor. If used in online mode you will get a dialog asking you in which view the object should be opened.

7.4.6. Edit Object with

This command works like *Edit Object*, but must be used in devices that can be opened in more than one editor.

7.4.7. Edit Object (Offline)

To open the object in its editor, it is necessary to be in online mode. Once in online mode, you can edit the object and apply changes using the *Online* > *Online Change* or *Online* > *Download* menu commands to transfer the modifications to the controller.

7.4.8. Project Information

Symbol: 

This command opens the *Project Information* dialog, where you can view and define properties and information on the project file, e.g. access attributes, version number, author and company information as well as Statistics concerning the project objects. Notice the possibility of external access on project information data via property keys and automatically generated functions

Four tabs are available for the information categories: *File*, *Summary*, *Properties* and *Statistics*.

7.4.8.1. File

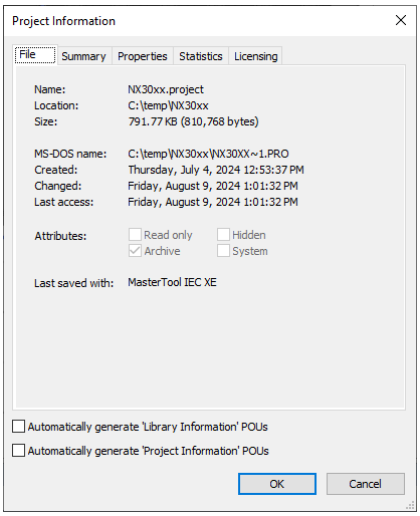

Figure 96: Project Information Dialog, File

Here the following properties of the project file are displayed: *Name*, *Location*, *Size*, *MS-DOS name*, *Created*, *Changed*, *Last access* and *Last saved with*.

Further on the currently set file attributes are shown: *Read only*, *Hidden* (by default not visible in Explorer), *Archive* (ready for archiving), *System* (system file), which by default are not editable in this dialog (see file properties in the *Windows Explorer*).

7.4.8.2. Summary

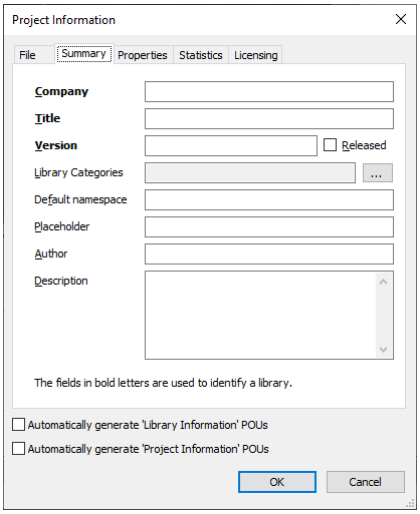

Figure 97: Project Information Dialog, Summary

Here you can optionally add some information on the project file like a *Company*, *Title*, *Version*, *Library Categories*, *Default namespace*, *Placeholder*, *Author* and *Description*. This information will automatically be displayed as available *keys* in the *Properties* tab of the *Project Information* dialog.

Note: If a project is intended to be used as a library in other projects, you must enter a Title, a Version number and the Company name here. Any library provided with these project information can be installed on the system and get included in a project. The company name besides the category serves for sorting in the *Library Repository* dialog.

Optionally also a Default namespace, the author's name and a short description can be specified in order to get saved as library project info. If no default namespace is defined here, automatically the name of the library project will be valid as namespace.

Assigning Library categories: The assignment of a library category later serves for sorting the available libraries in the *Library Repository* dialog. If no category is specified explicitly, the library will be added to category *Miscellaneous*. All special categories must be defined in a category description file in XML format. One or several category description files can exist for this purpose. You can call such a description file in order to choose the desired category for the local library project, or - alternatively - you can call another library project, which itself already has included the information from a category description file.

Via button ... open the dialog Library categories, which shows the currently assigned categories.

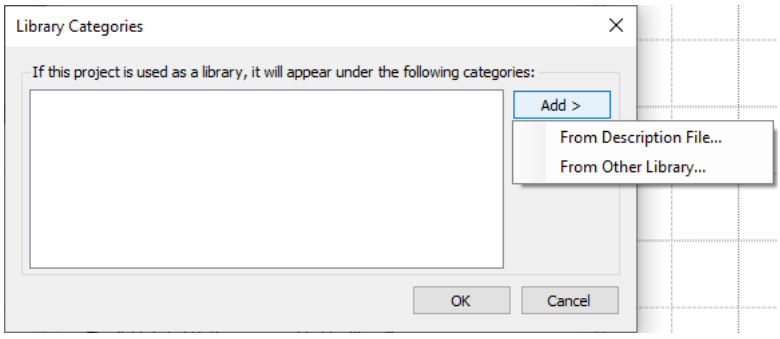

Figure 98: Library Categories Dialog

In order to assign the current library project to one or several (further) categories, use button Add and select one of the options.

- *From Description File...*: The standard browse-dialog for selecting a file will open where you can search for the desired description file *.libcat.xml.

- *From Other Library...*: The standard browse-dialog for selecting a file will open where you can search for a library already containing category information, *.library

The categories read from the description or library file will be listed now. Remove those you do not need by button *Remove*. Further categories might be added in the same way and finally you confirm with *OK* to get the dialog closed and to get the categories entered in the *Library Categories* field in the *Project Information* dialog.

7.4.8.3. Properties

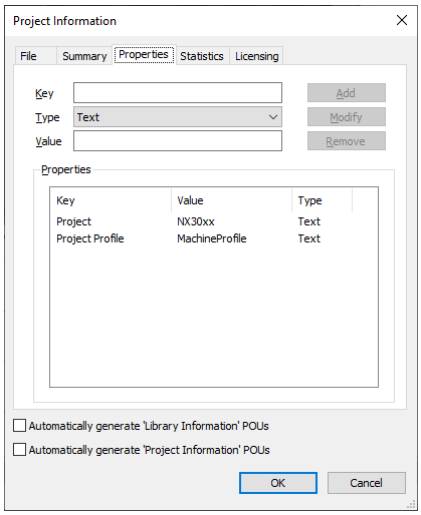

Figure 99: Project Information Dialog, Properties

Here you can define keys for some project properties. These later can be used in customer specific external programs for the purpose of controlling the respective property.

At least the information defined in the *Summary* tab of the dialog will be visible as *Keys* in the properties table. The properties names are used as key names, the data type automatically will be *Text* and the *Values* will be the text strings as defined in the *Summary* tab. Any further keys can be added as desired.

To add a key: Enter a key name in the *Key* edit field, choose the desired data type from the selection list at *Type* (Text, Date, Number, Boolean, Version) and in the *Value*: edit field enter the desired value, which must fit to the chosen data type. Press button *Add* to add the new key to the *Properties* table.

To modify a key: Select the entry in column *Key* of the *Properties* table, then edit the entries in the edit fields above the table and press button *Modify* to update the entry in the *Properties* table and for the respective keys also in the *Summary* tab.

To remove a key: Select the entry in column *Key* of the *Properties* table and press button *Remove*.

Automatically generate *Project Information* POUs: If this option is activated, automatically function POUs will be created in the POUs window, which can be used to access the project properties values in the application program. Special functions will be created in this case for the properties Company, Title and Version (GetCompany, GetTitle, GetVersion). For accessing additionally defined properties, a respective function for each property type (GetBooleanProperty, GetNumberProperty, GetTextProperty, GetTextProperty2, GetVersionProperty) will be available. In this case call the appropriate function, pass the property key (as defined in the properties tab) as an input, and you will get returned the property value.

Automatically generate *Library Information* POUs: If this option is activated, automatically function POUs will be created in the POUs window, which can be used to access the project properties values in the application program. Special functions will be created in this case for the properties Version and Released (GetLibVersion, GetLibVersionNumber, IsLibReleased).

7.4.8.4. Statistics

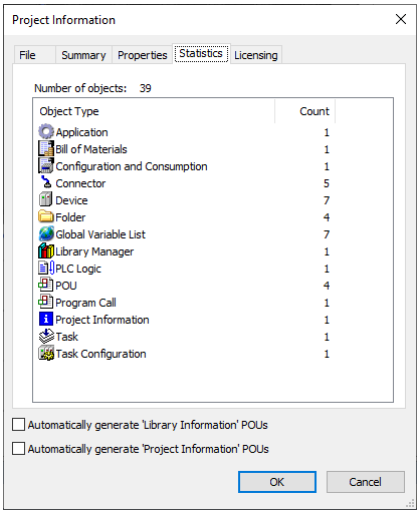

Figure 100: Project Information Dialog, Statistics

The table shows an overview on the objects used in the project: See the total *Number of objects* on top as well as in the table below the number of objects (*Count*) per *Object type*.

7.4.8.5. Licensing

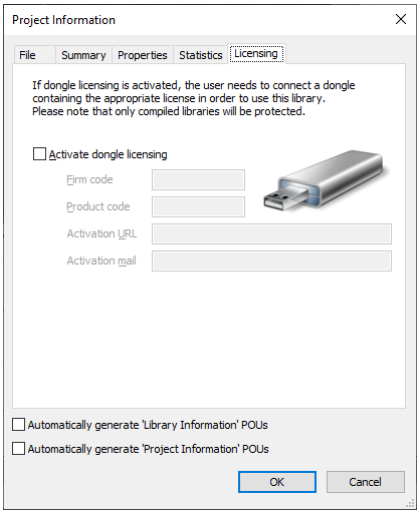

Figure 101: Project Information Dialog, Licensing

The dialogue concerns the protection of library licenses.

Activate dongle licensing: The library requires a dongle with a license for usage.

Firm code: License information that must be provided by the dongle for later use of the library.

ATTENTION

This method can only be used to protect *compiled libraries*.

7.4.9. Project Settings

Symbol: 

This command is available in the *Project* menu, and it is automatically included in the POUs tree.

The dialog provides subdialogs for various settings like for example project encryption, user and access rights management, version handling, layout definitions for printouts etc.

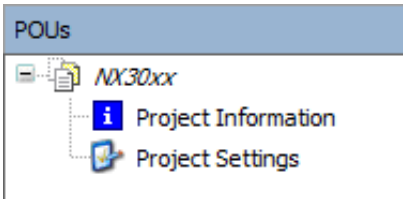

Figure 102: Project Settings Object in POUs Window

On the left side of the dialog the categories of possible settings are listed. On the right side the corresponding dialog will appear.

7.4.9.1. Compiler Warnings

Warning list whose verification in a project compiling is selectable.

7.4.9.2. Page Setup

Configures the printout layout. For further information, see Page Setup.

7.4.9.3. Source Download

This command is available for creating and transferring an archive file of the actual project to any device.

The command opens the *Select Device* dialog, where you have to choose the network path to the PLC like in the *Communication Settings* dialog. Select the appropriate entry in the tree of available devices and press *OK*.

This will set up a connection to the device as long as the source code gets downloaded in the form of an archive file.

The source code can be re-load to the programming system in offline mode by using command *Source upload*.

The default settings concerning destination device, content and timing for the source download are defined in the *Project Settings*, category *Source Download*.

7.4.9.4. Compile Options

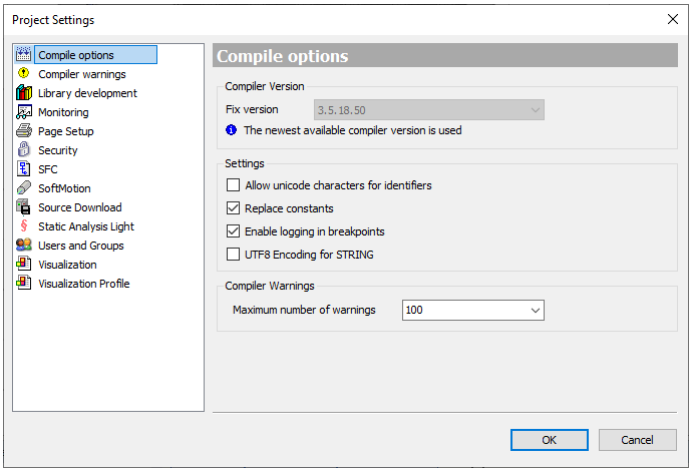

Figure 103: Project Settings Dialog, Category Compile Options

- *Fix version*: The compiler version used for both compiling and loading is displayed. Updating to the latest version is possible, but reverting to an older version is not.
- *Allow Unicode characters for identifiers*: Disabled by default, as the IEC standard doesn't allow Unicode characters in identifiers.
- *Replace constants*: Enabled by default. If enabled, scalar constants are loaded directly (excluding STRING, ARRAY, or structures). In online mode, constants are marked with a symbol before the value in the editor or monitoring view, restricting access (e.g., ADR operator, forcing, writing). Disabling allows access but increases computation time.
- *Enable logging in breakpoints*: Enable by default. For execution point breakpoints, you can create a message in the *Execution Point Settings* dialog. This message is logged when the application halts at the execution point.
- *UTF8 Encoding for STRING*: Disabled by default. If disabled, STRING data type is encoded in ASCII format project-wide, with proper index access to literals. If enabled, STRING is encoded in UTF-8 format, applied to all STRING literals and used for monitoring. Index access to UTF-8 literals is not recommended and may cause errors.
- *Maximum number of warnings*: Refers to warnings displayed in the message view.

7.4.9.5. Visualization Profile

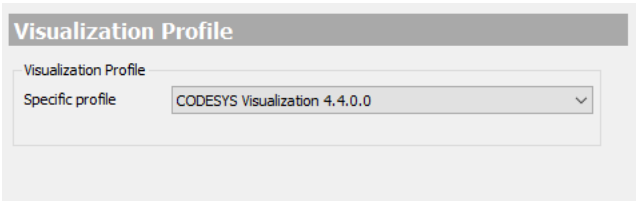

Figure 104: Project Settings Dialog, Category Visualization Profile

Define here a visualization profile to be used when the project gets opened. A **Profile** is the description of the a serie of *Visualization Elements*, present in the *Visualization Toolbox*. Currently there are only the *CODESYS VISUALIZATION 4.4.0.0* available.

7.4.9.6. Security

7.4.9.7. SFC

In the *Flags* dialog, it displays a list of variables reserved for use in SFC. In the *Build*, there a checkbox called *Generate active transitions only*, that if activated, generates code only for the transitions that are currently active.

7.4.9.8. Users and Groups

The *Project Settings* dialog in category *Users and Groups* provides three subdialogs for the user management for the current project:

- Users
- Groups
- Settings

For further information, consult User and Access Right Management

7.4.9.9. Visualization

The dialog used to configure the settings for the *Visualization* objects throughout the entire project. In this dialog there are two tabs, the *General* tab and the *Symbol Libraries* tab. You can see more specific explanations of this tabs with the figures below:

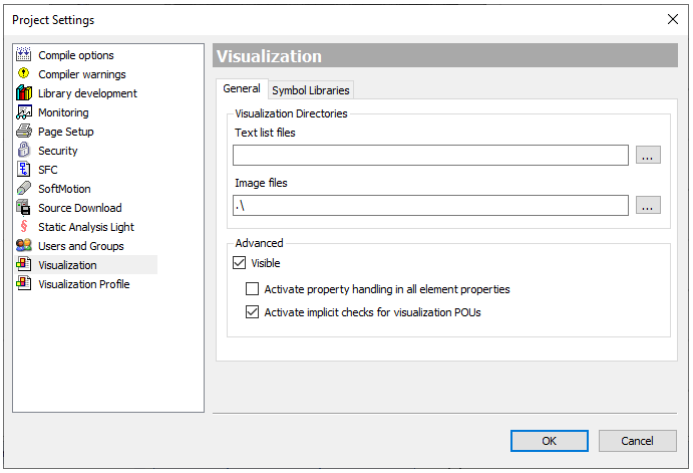

Figure 105: Project Settings Dialog, Category Visualization General Tab

- *Text list files*: Directory containing text lists for configuring multi-language texts in the project. Used by MasterTool IEC XE for tasks like importing or exporting text lists. Use the ... button to select the directory.
- *Image files*: Directory containing image files available in the project. Multiple folders are separated by a semicolon. Used by MasterTool IEC XE for tasks like importing or exporting images. Use the ... button to select the directory.

If *Visible* (disabled by default) is enabled, the following checkboxes can be selected:

- *Activate property handling in all element properties*: You can configure a visualization element with a property in properties that allow IEC variable selection. MasterTool IEC XE will then generate additional code for property handling during compilation. The IEC code must include at least one Interface property object type.
- *Activate implicit checks for visualization POUs*: If enabled, the implicit check also applies to visualization POUs, generating additional code that increases memory usage. If memory is limited, this option should be disabled.

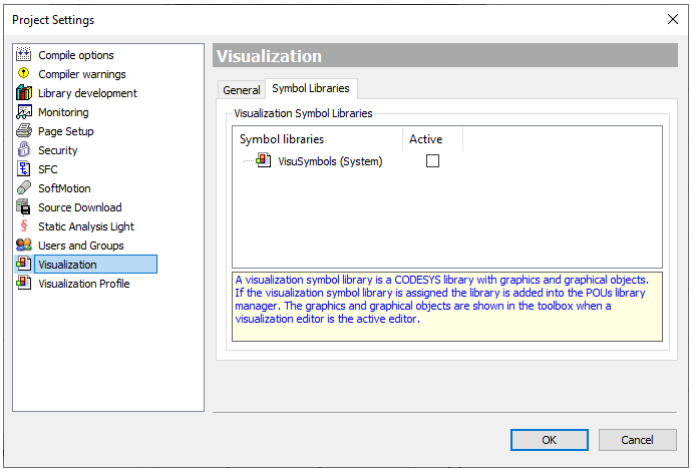

Figure 106: Project Settings Dialog, Category Visualization Symbol Libraries Tab

This dialog is the list of all installed symbol libraries, in MasterTool IEC XE case, only *VisuSymbols*. If the *Active* checkbox is enabled, the library will be available in the *Visualization Toolbox* view of a visualization. Otherwise, the library will be installed, but will not be available in the *Visualization Toolbox*.

7.4.9.10. SoftMotion

In this dialog it's possible to set the SoftMotion version for the project.

7.4.9.11. Static Analysis Light

The dialog triggers the checks conducted by the light version of CODESYS Static Analysis whenever code is generated.

7.4.9.12. Monitoring

In online mode there are various possibilities to display the current values of the watch expressions of an object on the PLC:

- Inline monitoring in the implementation editor of an object. For details see the description of the respective editor.
- Online view of the declaration editor of an object. For details see the description of the Declaration Editor.
- Object-independent watch lists. For details see the description of the watch views.
- *Trace* sampling. Recording and display of variable values from the PLC. For details see the description of the Trace functionality.

Note: In online mode there is a limitation of 25000 entry variables monitorable in POUs edited with the ST editor, the user will be alerted when the limit is exceeded with a build error.

7.4.9.13. Library Development

This dialog inform the definitions inside the libraries. The *SCAN* button read the libraries instantiated in the project.

7.4.10. Project Update

This command opens the window Project update, it aims to allow modifying the device used and the current project profile.

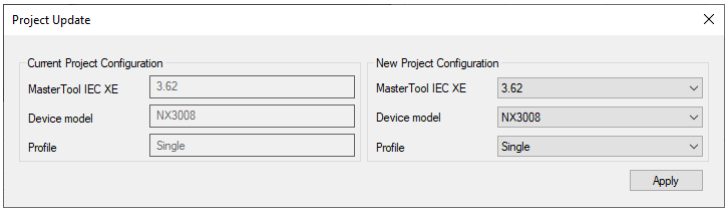

Figure 107: Project Update

7.4.10.1. Modify Device

Allows updating projects created with previous versions of MasterTool IEC XE and modify the CPU model and profile. In this group are the following fields:

- *MasterTool IEC XE* : Display the product version.
- *Device model*: Display the model of the device.
- *Profile*: Display the profile of the project.
- *Apply*: Apply the changes on *New Project Configuration*.

Note: If the model of CPU used is modified, this name is not changed. This behavior may cause confusion if the user does not change the default name, which is the CPU model with which the project was created. For example, a project created with NX3008 CPU with it's default name kept, after the CPU modified to NX3020 will keep your name as *NX3008*, until it is modified by the user.

Note: This operation can be slow and may cause loss of devices configuration. Before the process is started, the user can create a project file from the original project.

7.4.10.1.1. Updating an Old Project

When a project was created in a different MasterTool IEC XE version from that one installed on the computer, it is necessary to make a modification to the device available version in the updated product. Accordingly, the *Device* object in the configuration tree project will display the icon at its side, as well as other objects that were also modified from one version to another.

Update should always be taken to a more current version of the device but of the same type, i.e. , if the project was created with a NX3008 CPU model the upgrade should be made to the most current version of this same model.

Whenever a project is to be updated, the user is asked if they really want to update that project. They are also asked if they want to save a backup of the project, as the update cannot be undone.

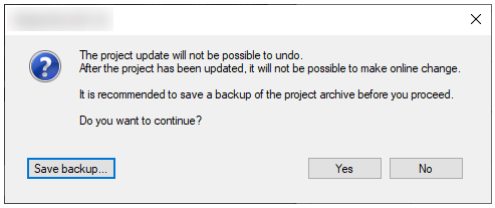

Figure 108: Project Update Save Backup

7.4.10.1.2. Old Projects with PROFIBUS Slaves from Other Vendors

To use PROFIBUS Slave from other vendors that aren't from Ponto Series and Nexto Series, it's necessary to install the device description file (GSD) using the Tools menu. Once the GSD is installed, it can be used in other projects created with the MasterTool IEC XE .

When a MasterTool IEC XE version is uninstalled, it removes all installed files, including the device description files installed with the GSD. This way, by uninstalling MasterTool IEC XE version and installing a new one, if projects with slaves from other vendors created prior to that are opened, they will be displayed in the project tree with the icon. In these cases, two actions can be taken in order to correctly use the project:

- Manually install all GSDs before opening the project.
- Before uninstalling the MasterTool IEC XE , create a project archive, through the *File* > *Project Archive* > *Save/Send File...* menu, with all the GSDs installed and used in the project prior to the uninstalling. The devices present in the project archive are going to be installed automatically in MasterTool IEC XE device repository.

If the project is opened without the device installed, a message with the text *Internal error while object 'Device' is providing language model information: stDisplayName* are going to be shown during the project opening.

If a Project is created in a MasterTool IEC XE prior to 2.00, even if the procedure described above is executed, the icon will be displayed for the PROFIBUS slaves from other vendors. This happens because the devices import mechanism was changed in version 2.00. After the device installation with the GSD or project archive, it's necessary, for these cases, to remove the device and add it again, redoing its configuration.

7.4.10.2. Modify Project Profile

Modifies the project profile currently used in the project and apply the rules defined for the new profile design. It includes the following fields:

- *Current profile*: displays the profile currently set in the project.
- *Select profile*: displays all the options available for modifying the project profile.
- *Modify*: performs the modification and apply the profile of the project selected by the user to the project.

7.4.11. Document

Symbol: 

This command opens the *Document Project* window, where you can configure and print a documentation for the project.

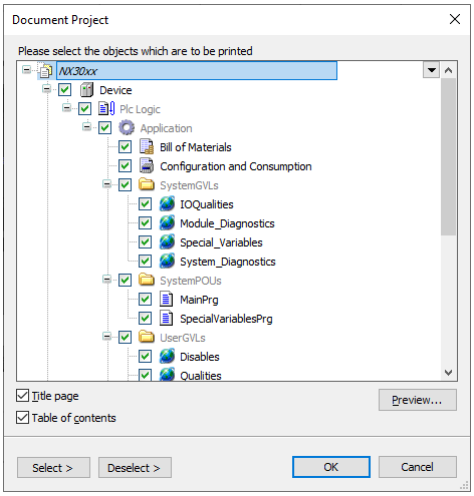

Figure 109: Document Project

The project objects in POUs windows and devices are presented in a tree and the user can select which object(s) must be documented in printed form. The contents of objects displayed in gray are not printable, but it will be considered in the documentation to the extent that the name of the object is inserted in the structure and content of the tree.

For sorting and searching in the object tree a toolbar is available above the tree.

If *Title page* option is selected, the documentation will be increased to more than one page to cover. If *Table of contents* option is selected, the documentation will be increased by a content page showing all objects and the corresponding page numbers in the documentation. The header levels are defined in the *Page Setup* window.

The buttons *Select* and *Deselect* shows options to select/deselect all or all type of a determined object.

7.4.12. Compare

Symbol: 

By use of the command *Compare* you may compare the actual project with another one (reference project). The reference project and the compare options are specified in the *Project Compare* dialog, which will be opened, when the command is performed.

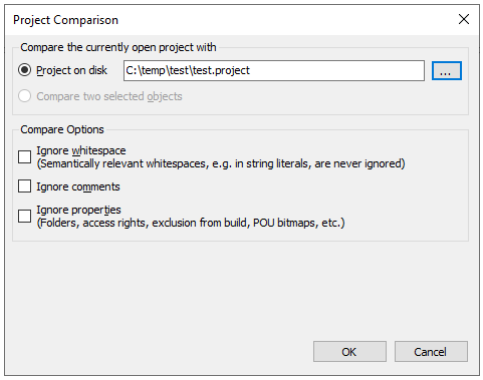

Figure 110: Project Compare Dialog

On *Compare the currently open project with* field, the reference project can be:

- *Project on disk*: By default the path of the actual project itself is entered, so that it will be compared against its version latest saved. You can replace it by modifying the file path after a click on the text field. Alternatively, you may make use of the ... standard dialog box for browsing after a click on.

On *Compare* options field one or several of the following options concerning the comparison can be activated:

- *Ignore whitespace*: Discrepancies due to a different number of blanks will not be mentioned.
- *Ignore comments*: Comments will be excluded from comparison.
- *Ignore properties*: Object properties will not be compared.

After closing the dialog *Project Compare* with *OK*, the comparison is executed according to the settings.

7.4.12.1. Survey of Comparison Result by means of Marked Device Trees

The result of the comparison is represented in a new window entitled *Project Comparison - Differences*.

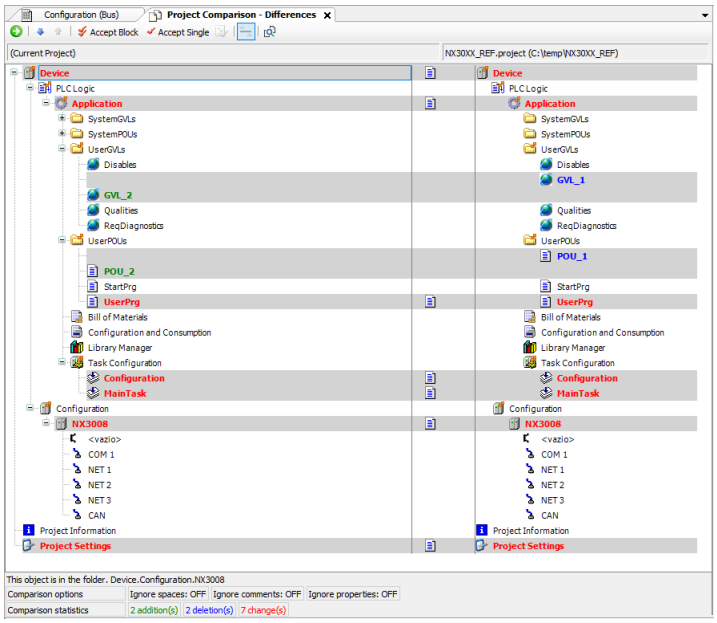

Figure 111: Comparison Result

On top of the new window a toolbar is placed at your disposal, followed by a title bar and a sub-window. The title bar as well as the sub-window itself is subdivided into a left part representing the current project and the right one representing the reference project. The corresponding file paths are displayed in the title bar whereas the sub-window shows the two device trees associated to the projects.

Therein, you see the name of identical units displayed in black with no further remarks. Otherwise, the name of an unit is displayed:

- In bold blue: if it exists in the reference project only; instead of its name a gap will be inserted at the corresponding place in the device tree of the actual project.
- In bold green: if it exists in the actual project only; instead of its name a gap will be inserted at the corresponding place in the device tree of the reference project.
- In bold red: in both parts of the window, if there are differences in the two versions of the unit concerning the implementation. In addition the unit name is followed by .
- In red: if the differences of the versions are related only to properties —- or access rights .

It's possible to navigate through the modified or new objects using <CTRL>+<DOWN> or <CTRL>+<UP>. The command can be seen in the tool bar in the tab. The *Accept Block* ( ) command allows you to select a block, including all subordinate objects and units, for acceptance from the reference block to the current block. Meanwhile, the *Accept Single* ( ) command enables the selection of an object within the current object for acceptance from the reference line.

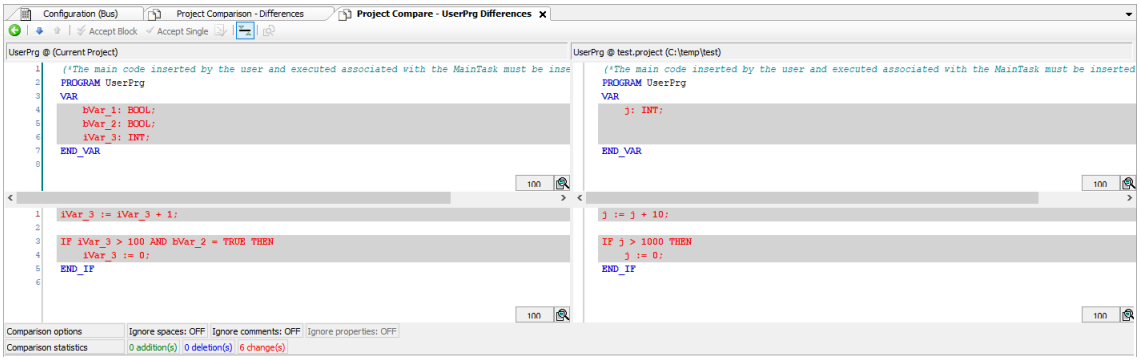

Figure 112: Example of Detailed ComparisonResult

7.4.13. Commit Accepted Changes

Symbol:

This command will actually apply what was previously accepted in the *Compare* command.

7.4.14. Export PLCopenXML

The PLCopen is an independent entity of manufactures or brands that promote the use of international standards in automation, especially in those cases related to programming and the languages defined in IEC 61131-3 norm. It promotes it through certifications and also through other standards that are based in the IEC 61131-3 norm.

The PLCopenXML is a standard that promotes a manufacturer independent interface to store the information of a project created using the concepts of the IEC 61131-3 norm. This standard predicts a XML file format with schemes determined to store each structure, data and configuration type predicted in IEC 61131-3. These files can be used in any tool that supports that norm.

The Export PLCopenXML command allows to export the objects of a project created with MasterTool IEC XE in a PLCopenXML format. By running this command, the screen in the figure below will be displayed.

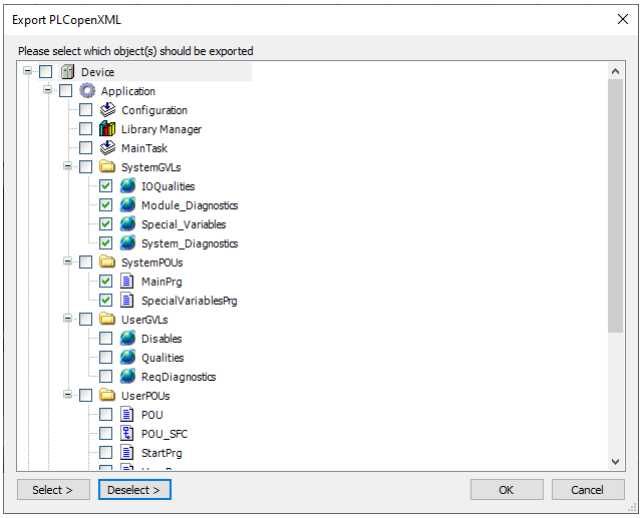

Figure 113: Export PLCopenXML Screen

In this screen it's possible to configure individually all objects that are going to be exported. When an object that is parent to others, in the tree structure of the project, is marked or unmarked, all of its children are going have that same action upon them.

To ease editing it's also possible to use the *Select* and *Deselect* buttons. Through them it's possible to select all objects of a certain type present in the projects, to them mark or unmark them.

After selecting the objects to be exported and press the *OK* button a default save window from the operating system will be displayed. In it, the file name and the path to save it must be chosen. The file is saved with *.xml extension.

7.4.15. Import PLCopenXML

The command *Import PLCopenXML* allows to import into the project the objects from a file in the PLCopenXML format created with the MasterTool IEC XE or other tool that can save files in the PLCopenXML format.

By executing the *Import PLCopenXML* a default *open* window from the operating system will be displayed. The file to be imported must be selected and it must have the *.xml extension. After that, a screen such as the one in the figure below is going to open.

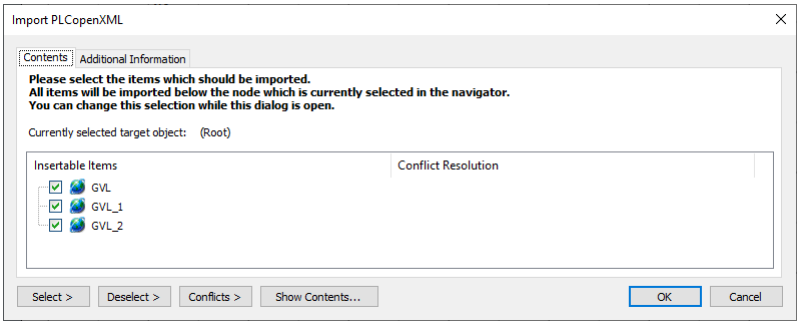

Figure 114: Import PLCopenXML Screen

In this screen it's possible to select which objects available in the file will be added to the project. These objects can be marked individually or with help of the Select and Deselect buttons that allows to select all the objects of a certain type present in the file to be marked or unmarked. When an object that is parent to others, in the tree structure of the project stored in the file, is marked or unmarked, all of its children are going have that same action upon them.

For the content to be correctly shown it's necessary that, before executing the Import PLCopenXML command, the parent object is selected in the project tree. For example, if POUs and GVLs were saved, declared below the Application object, this object must be selected before executing the import command. If this is not done, the Contents tab will appear empty. Yet the Show Contents button can be pressed and will display what is saved in the file.

The *Additional Informations* tab presents information about the saved file, e.g. the tool in which the file was created.

By clicking in the *OK* button, all selected objects will be added to the project.

7.4.16. Restore Points

Symbol:

The *Restore Points* is a command used to create time points on the project where the user can navigate through it. To create a restore point, go to the *Project* tab > *Restore Points* and click the *Create* button, as shown in the image below:

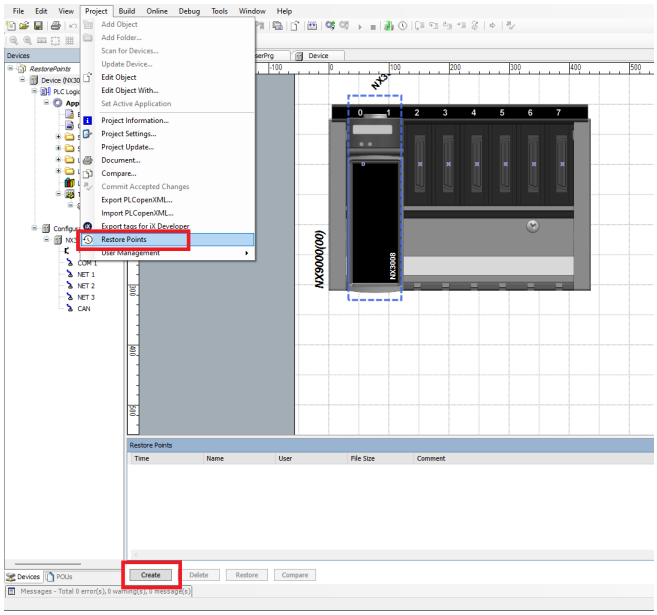

Figure 115: Accessing Restore Point

The following dialog will appears in the screen:

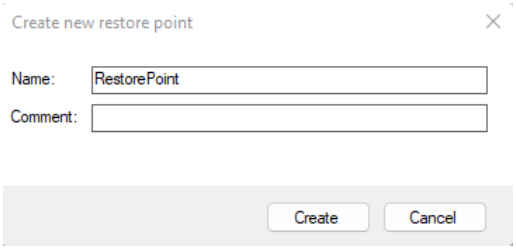

Figure 116: Create New Restore Point

After clicking the *Create* button, the restore point will be added to the list, as shown in the figure below.

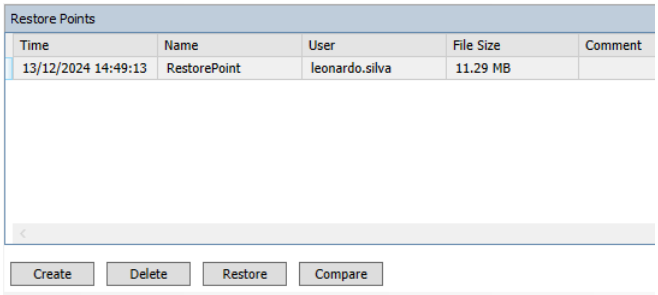

Figure 117: Restore Point Created

This will give you the chance to change the name of the restore point and write a comment. In the *Restore Point* dialog there will be five columns with data about it:

- *Time*: The moment the restore point has just been generated.
- *Name*: The restore point name.
- *User*: The computer user of the person generating the restore point.
- *File Size*: The size of the file generated.
- *Comment*: Where user comments can be inserted.

To restore a point, click in the *Restore* button, the following dialog will appear asking if you want to proceed, and if you want to create a new restore point:

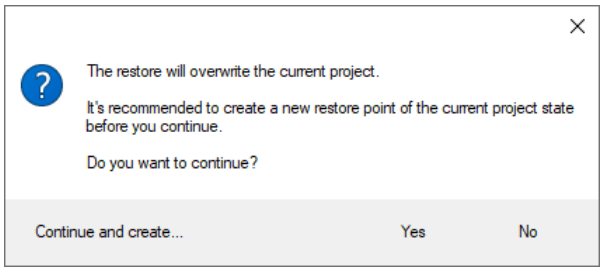

Figure 118: Restore

If you accept, that point will be restored. The *Compare* button compares the actual project with the point selected in the dialog. Clicking in the this button, the following dialog appears:

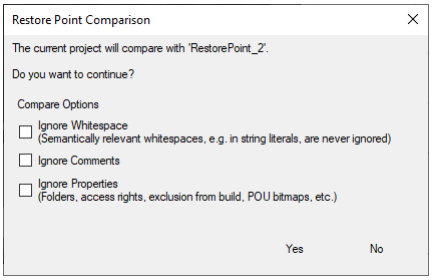

Figure 119: Restore Point Comparison

To delete a restore point just select it and click in the *Delete* button (or the <DEL> key).

7.4.17. Import Safety Project...

Import a safety CPU and the objects in it's project to MasterTool. Uses the project (*.project) file itself.

7.4.18. Delete Safety Objects Imported...

Delete all the safety objects imported to MasterTool.

7.4.19. User Management

The *User Management* category provides commands for configuration of access rights on the project objects logging on or off to/from the project via a defined user account in order to get the access rights which are associated to this account

The configuration of user accounts and groups is done in the *Project Settings* subdialog *User Management*. For an overview see User and Access Right Management.

Available commands:

- User Login..
- User Logout
- Permissions..

7.4.19.1. User Login...

This command opens the *Log In* dialog for log in to a project or library via defined user account. Logging on with a certain user account means to log on with those object access rights, which are granted to the group, which the user belongs to. The configuration of user accounts and groups is done in the *Project Settings* subdialog *User Management*. For an overview see User and Access Right Management.

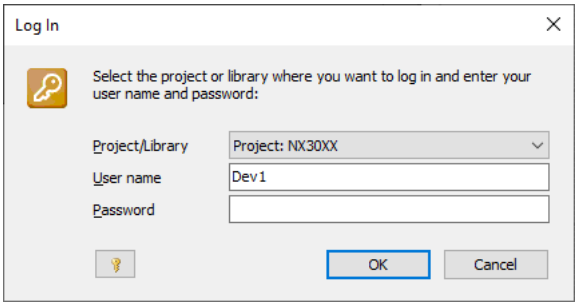

Figure 120: User Login

To log on select the project or an included library from the selection list in the *Project/Library* field. Enter *User name* and *Password* of a valid user account, noticing that each project or library has an own user and access rights management. Log on with *OK*.

If already another user is logged on the project, this one will be logged out automatically by the new log in action.

When you are logged on to a project or library and try to perform an action for which you have no right, automatically the following *Log In* dialog will be opened, giving the possibility to log on with another user account provided with the appropriate rights:

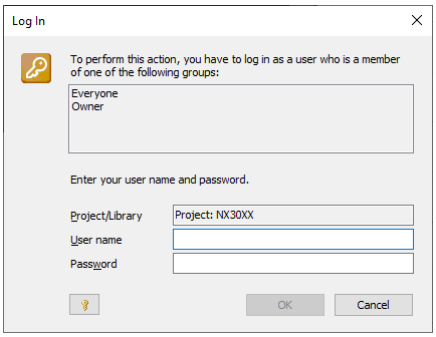

Figure 121: Log In Dialog on a Non-Permissible Action

The upper part of the dialog shows - just for information - all groups which are provided with the necessary rights for the desired action. If you have an user account for one of these groups you now might log-on with the appropriate user name and password and finally perform the desired action.

The status bar always displays which user currently is logged on the project.

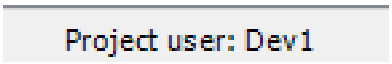

Figure 122: Status Line

7.4.19.2. User Logout

Symbol: 

This command logs out the currently logged-in user from the current project. No dialog or message appears.

If no user had been logged on to the currently opened project or to a referenced library, an appropriate message will appear when trying to logout

If the user currently is logged in to more than one project or referenced library (not necessarily with the same user account) a Logout dialog will appear. From the Project/Library selection list choose those project/library for which you want to log off. The name of the Current user is displayed just for information.

The status bar always displays which user currently is logged on the project.

7.4.19.3. Permissions...

This command opens the *Permissions* dialog, where the rights to work on objects or to perform commands in the current project can be configured. Notice that the rights concerning objects also, (same effect) might be configured in the object *Properties* dialog.

For further information check User and Access Right Management.

7.5. Recipe Menu

Visible only when the editor of the *Recipe Definition* object is active, the *Recipe* menu provides commands to work with variables that were added in the *Recipe Definition* object.

For more information about the *Recipe Definition*, see Recipe Manager.

7.6. Menus from the Editors of Programming Languages

As the POU language being edited is enabled a menu with the options corresponding to this language. Are three different menus:

- FBD/LD Editor Commands
- CFC Commands
- SFC Commands

The ST language editor does not have a specific menu.

7.6.1. FBD/LD Editor Commands

The *FBD/LD* category provides commands for working in the FBD/LD Editor, which is a collective editor for the languages FBD (Function Block Diagram) and LD (Ladder Diagram).

7.6.1.1. Insert Network

Symbol: 

Default Shortcut: <CTRL>+<I> or <CTRL>+<T>

This command is used to insert a network in the FBD or LD editor.

If the cursor currently is placed within an existing network, the new network will be inserted immediately above that network. If the cursor currently is placed in the editor window but not in a network, the new network will be added at the end of the current network list. The network numbering will be updated automatically.

Note: The display of the elements in a LD/FBD/IL network is defined in the FBD, LD and IL Editor.

7.6.1.2. Toggle Network Comment State

Symbol: 

Default Shortcut: <CTRL>+<O>

This command can be used in FBD or LD editor to comment out a network to set it back from comment to normal state. The command will affect the network in which the cursor is currently positioned.

A commented-out network will be displayed according to the options set for comments and will not be noticed in program processing.

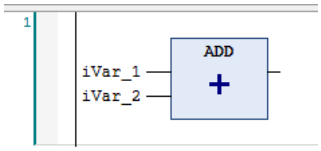

Figure 123: Network Normal State

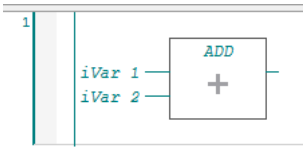

Figure 124: Network Comment State

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor.

7.6.1.3. Insert Assignment

Symbol: -[VAR]

Default Shortcut: <CTRL>+<A>

This command is used to place an assignment in a LD or FBD.

Depending on the selected position insertion takes place directly in front of a selected input, directly after a selected output or - if a whole network or sub-network is selected - at the end of the network or sub-network.

In FBD the assignment is inserted as a line followed by *???*, in LD it is represented by a coil and *???*.

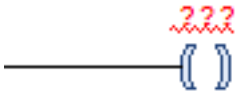

Figure 125: Assignment in LD

Figure 126: Assignment in FBD

7.6.1.4. Insert Box

Symbol:

Default Shortcut: <CTRL>+<B>

This command is used to insert a box element into a network for the purpose of calling an operator, a program, a function block, a function or an interface.

As soon as you choose the command, the *Input Assistant* dialog will open, providing the appropriate categories of POUs. Select one and confirm with *OK* to insert the box at the currently selected position in a network resp. to create the corresponding IL instructions.

Alternatively, you can choose command *Insert Empty Box* so that you can enter the desired box type directly. A very comfortable way to add a box is to drag it directly from the *Toolbox* or from another position within the editor.

7.6.1.4.1. FBD or LD Editor Specific Characteristics

Boxes of type program or function block always are inserted in line, that is the processing line will be connected to the uppermost input and output of the inserted POU.

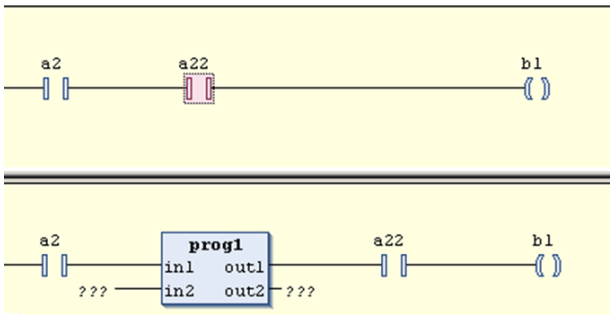

Figure 127: Insert Function Block

The text within the box shows the box type (for example F_TRIG) and is editable. By replacing this text by the type name of another valid module, the user can replace the box by another one. An existing box also can be replaced by inserting another one at the same position. Notice that if already inputs have been defined for the previously used box, these will be kept, except the new box has a lower maximum number of inputs. In this case, the last inputs will be deleted accordingly.

If provided with the respective module and if option *Show box icon* is activated, an icon will be displayed within the box.

Within parallel connections in a LD network no insert positions will be offered when dragging a box element from the *Toolbox*. Reason: A POU call (box) needs a direct connection to the power rail.

In the LD editor boxes for the call of certain operators automatically are inserted with EN and ENO resp. only EN in- and outputs. EN and ENO connections get those, which have a non-boolean output (for example ADD, SEL, BOOL_TO_INT), only an EN input get those, which have a boolean output (for example EQ, GE, GT). At a box with an ENO output it is not possible to insert a further box at the other outputs of this box.

VAR_IN_OUT parameters of an inserted POU box are marked with a bidirectional arrow.

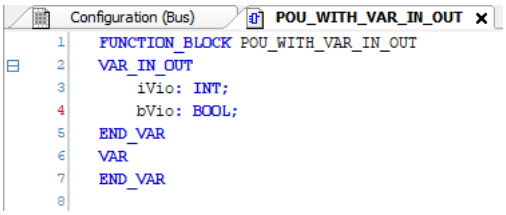

Figure 128: POU_WITH_VAR_IN_OUT

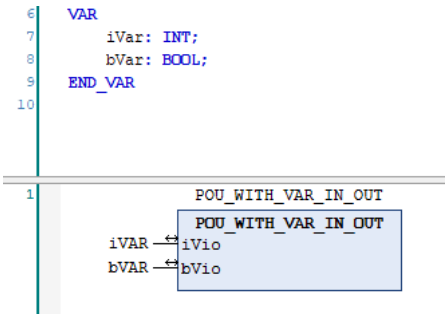

Figure 129: POU_WITH_VAR_IN_OUT Instantiated

Function block boxes have an editable field above the box where you have to enter the name of the instance variable. If a box representing the instance of a function block instance gets replaced by inserting another function block type, the instance has to be redefined also.

In functions and function blocks, the formal names of the in- and outputs are displayed. The main output of the function (return value) however is displayed without name.

In case the box interface has been changed, you can update the box parameters (for example modified number of outputs) by command *Update Parameters*. The update will not be done automatically.

Concerning insert positions, the most recently inserted POU will be inserted at the currently selected position:

- Input: the box will be inserted before this input. Its first input and - if applicable- its first output will be linked within the existing branch.
- Output: the box will be inserted after this output. Its first input and - if applicable- its first output will be linked within the existing branch.
- Another box: The old element will be replaced by the new POU. As far as possible, the connections will remain as they were before the replacement. If the old element had more inputs than the new one, then the unattachable branches will be deleted. The same holds true for the outputs.
- Jump or Return: The box will be inserted before this jump or return. Its first input and - if applicable- its first output will be linked within the existing branch.
- Network or Sub-network: The box will be inserted following the last element of the network or sub-network and be linked with its first input.

All box inputs that could not be linked will receive the text *???*. This must be selected and replaced by the name of the desired constant or variable.

If there is already a branch to the right of an inserted box, this branch will be assigned to the first output of the box. Otherwise, the outputs remain unassigned.

Note: Concerning the view options for the components of a LD/FBD network notice the settings in the FBD, LD and IL Editor

7.6.1.5. Insert Empty Box

Symbol:

Default Shortcut: <CTRL>+<SHIFT>+<B>

This command is used to insert an empty box element into a network.

In contrast to *Insert Box* command, when inserting an empty box not automatically the *Input Assistant* will open, but the instance field above the box.

Then decide, which type of box you need:

- If the box should represent a function block, then enter the desired instance variable name and close the input with <ENTER>. You might use the *Input Assistant* ... for entering the name of an already existing instance. After inserting an already declared instance variable the function block immediately will be displayed accordingly. If you entered a not yet known instance name, you also must specify the name of the desired function block. For this purpose after closing the instance name input, the input focus automatically changes to the box type edit field within the box. In this case the input assistant will only offer function blocks.
- If the box should represent an operator, program, function or interface, then when the cursor is placed in the instance field, just press the down arrow key. The input focus will change to the box *Type Field* (within the box), where you can enter the respective operator, program, function or interface name, directly or via the Input Assistant. After having terminated this input, the box will be displayed accordingly.

7.6.1.6. Insert Box with EN/ENO

Symbol:

Default Shortcut: <CTRL>+<SHIFT>+<E>

This command is used to insert a box element with the insertion of EN and ENO input and outputs in a network. Besides this difference, this command work in a similar way as the *Insert Box* command.

7.6.1.7. Insert Jump

Symbol:

Default Shortcut: <CTRL>+<L>

This command inserts a jump. The target of a jump is another network that is the label of that network.

In FBD or LD, depending on the selected cursor position, insertion takes place directly in front of a selected input, directly after a selected output or - if a whole network or sub-network is selected - at the end of the network or sub-network.

For an inserted jump, a selection can be made accompanying the entered text *???*, and the jump can be replaced by the name of the label to which it is to be assigned.

Figure 130: Insert Jump

To a *Insert Jump* command work, it's necessary to have a *Label* of the same name in a network. It's for this label that the jump will occur. To have more information see Insert Label.

7.6.1.8. Insert Label

Symbol:

This command adds a label field to the selected network. You can customize the default text string to meet your specific requirements.

- You may enter the name of a label providing a target for a jump instruction.
- You may enter a pragma instruction. See **Pragma Instructions** (IEC 61131 Programming Manual) for a detailed description.

If you want to make use of both of the alternatives, enter the pragma first followed by the name of the label.

7.6.1.9. Insert Return

Symbol:

This command inserts a RETURN instruction.

7.6.1.9.1. FBD or LD

Depending on the selected position insertion takes place directly in front of a selected input, directly after a selected output, directly before a selected line cross or at the end of a network or sub-network.

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor.

7.6.1.10. Insert Input

Symbol: 

Default Shortcut: <CTRL>+<Q>

This command inserts an additional input at an extensible box (AND, OR, ADD, MUL, SEL) in FBD or LD editor.

The maximum number of inputs depends on the box type (for example, ADD can have two or more inputs).

In order to extend a box by an input at a certain position, select the input above which you wish to insert an additional input.

In order to extend a box by an input at the lowest (last) position, select the box body.

The new input primarily is allocated with the text *???*. Replace this text by the name of a desired constant or variable. The Input Assistant might be used for this purpose.

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor.

7.6.1.11. Insert Coil

Symbol: 

Default Shortcut: <CTRL>+<A>

This command is used to insert a coil in parallel to the previous coils.

If the marked position is a connection between the contacts and the coils, then the new coil will be inserted as the last. If the marked position is a coil, then the new coil will be inserted directly above it.

The coil is given the text *???* as a default setting. You can click on this text and change it to the desired variable or you can use the *Input Assistant* instead.

For the possibility of entering addresses, of line breaks for variable names and of comments per coil please see the description of the FBD, LD and IL Editor.

7.6.1.12. Insert Set Coil

Symbol: 

This command is used to insert a set coil. Alternatively, the combination of the commands *Insert Coil* and *Extras Set/Reset* can be used to insert a set coil.

7.6.1.13. Insert Reset Coil

Symbol: 

This command is used to insert a reset coil. Alternatively, a combination of the commands *Insert Coil*, *Extras and Set/Reset* can be used to insert a reset coil.

7.6.1.14. Insert Contact

Symbol: 

Default Shortcut: <CTRL>+<K>

This command inserts a contact in a LD network. It is not available in the FBD editor but will be converted appropriately when switching views.

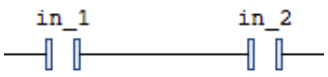

Figure 131: Insert Contact in LD

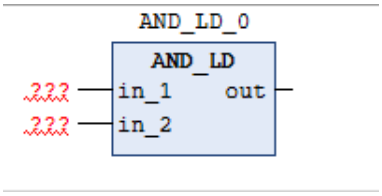

Figure 132: Visualization of Contact Inserted in FBD

The new contact will be inserted in line left to the currently selected contact or box. If the cursor position is within an existing parallel connection, the new contact element also will be inserted within.

Note: Alternatively, a contact element can be inserted by dragging from the toolbox or from another position within the editor. If however it should not be inserted within, but before, behind or between existing parallel connections, this only will be possible by using the menu command. For this purpose first in each of the branches of the parallel connection select one of the contact elements multiselection while keeping pressed the <CTRL>+<ARROW KEY> and then use the command.

7.6.1.15. Insert Negated Contact

Symbol:

This command is used to insert a negated contact. Alternatively, a combination of the commands *Insert Contact* and *Negation* can be used to insert a negated contact.

7.6.1.16. Insert Contact (right)

Symbol:

This command inserts a contact in a LD network. The same is valid as for command *Insert Contact*, except that the new element will not be inserted left but right to the current cursor position in line.

7.6.1.17. Insert Contact Parallel (below)

Symbol:

Default Shortcut: <CTRL>+<R>

This command inserts a contact parallel to the marked position in the network. The same is valid as for command *Insert Contact Parallel (Above)*, except that the new contact will be inserted below the selected position.

7.6.1.18. Insert Negated Contact Parallel (below)

Symbol:

This command is used to insert a negated parallel contact. Alternatively, a combination of the commands *Insert Contact Parallel (Below)* and *Negation* can be used to insert a negated parallel contact.

7.6.1.19. Insert Contact Parallel (above)

Symbol:

Default Shortcut: <CTRL>+<P>

This command inserts a parallel contact above the currently marked position in the network (parallel connection). The command is not available in the FBD editor but will be converted appropriately when switching views.

Notice that also multiple elements might be selected in order to get the new contact inserted parallel to those.

The contact is preset with the text *?*. You can click on this text and change it to the desired variable or the desired constant. The Input Assistant can be used for this purpose.

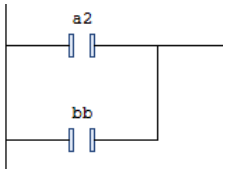

Figure 133: Contact Parallel (position was selected contact *bb*) inserted in LD

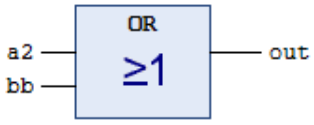

Figure 134: Visualization of contact parallel in FBD

7.6.1.20. Paste Contacts: Paste below

Default Shortcut: <CTRL>+<F>

This command is only available in LD editor. It pastes the elements or the section of the network, which has been put to the clipboard before by a *Copy* or *Cut* command, below the currently selected contact element in the network. This corresponds to the common *paste* command.

7.6.1.21. Paste Contacts: Paste right (after)

Default Shortcut: <CTRL>+<G>

This command is only available in LD editor. It pastes the elements or the section of the network, which has been put to the clipboard before by a *Copy* or *Cut* command, right to the currently selected contact element in the network.

7.6.1.22. Paste Contacts: Paste above

Default Shortcut: <CTRL>+<H>

This command is only available in LD editor. It pastes the elements resp. the section of the network, which has been put to the clipboard before a *Copy* or *Cut* command, above the currently selected contact element in the network.

7.6.1.23. Negation

Symbol: 

Default Shortcut: <CTRL>+<N>

This command is used in FBD or LD to toggle the negation of an input, an output, a jump or a RETURN instruction.

At boxes, jumps or returns the symbol for the negation is a small circle at the respective input or output connection.

A negated contact in LD is indicated by a slash in the contact symbol:

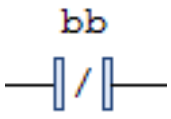

Figure 135: Negated Contact

To negate a contact or coil select the element (cursor positions 8 resp.9) and use the command.

Notice that the *Toolbox* in category *Ladder Elements* provides negated contact elements for inserting by drag & drop.

To negate an input or output put the cursor according to cursor positions 2 or 4.

To negate a jump select the last preceding output (cursor position 4).

To cancel the negation of an element use the same cursor positions and also perform the Negate command.

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor

7.6.1.24. Edge Detection

Symbol: 

Default Shortcut: <CTRL>+<E>

This command is used in FBD or LD to insert an edge detection element at a boolean input. This corresponds to inserting a R_TRIG function block for detecting a rising edge (FALSE > TRUE) respectively a F_TRIG function block for detecting a falling edge (TRUE > FALSE). When the command is performed repeatedly at the same insert position, the inserted element will toggle between rising edge detection (), falling edge detection () and none.

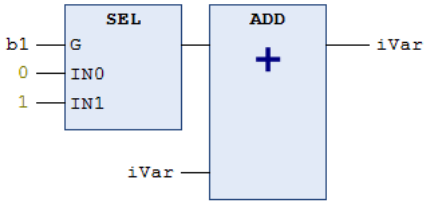

Figure 136: Edge Detection at SEL Operator

In this example an Edge Detection element has been inserted when the first input (b1) of the SEL box was selected. The SEL operator will have output *1* each time a rising edge is detected at its input.

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor.

7.6.1.25. Set/Reset

Symbol: 

Default Shortcut: <CTRL>+<M>

This command is used in FBD or LD to define the type of the boolean outputs (as *Set*, *Reset* or normal).

With multiple executions of the command, the output will alternate between set, reset, and normal output.

7.6.1.26. Set Output Connection

Symbol: 

Default Shortcut: <CTRL>+<W>

This command can be used in FBD or LD with boxes, which have multiple outputs to determine that output which should be connected to the network processing line.

Notice the shift of the output assignments in case of changing the output connection.

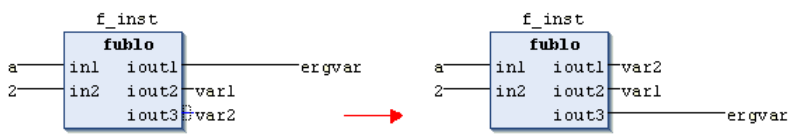

Figure 137: Change Output Connections

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor.

7.6.1.27. Insert Branch

Symbol: 

Default Shortcut: <CTRL>+<SHIFT>+<V>

This command branches the current execution line within a network in FBD/LD.

The current line will be split into two *sub-networks*: An additional line will be drawn *below* the existing one. In online mode the two subnetworks after a branching point will be executed one after the other from up to down.

If you drag the *branch* element from the toolbox or another position within the network, you will get indicated all possible insert positions by grey position markers.

If you drag the *branch* element from the toolbox or another position within the network, you will get indicated all possible insert positions by grey position markers.

A branch can be inserted at the input connectors of boxes which are not positioned in a sub-network, at the output connectors of a box if that is not connected (also indirectly) to the input of another box within a sub-network, at the connector between contacts and coils (cursor position 10), or at a contact. A branch cannot be inserted inside of *OR* in contacts groups and inside of multiple assignment groups.

Each subnetwork gets an own *marker*, an upstanding rectangle symbol, which serves for selecting the sub-network.

For information on cursor positions, see the corresponding item in IEC 61131 Programming Manual.

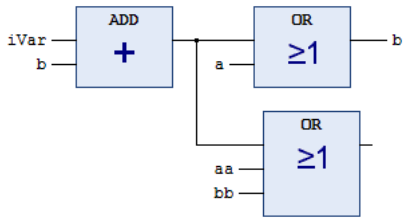

Figure 138: Sub-network Markers in FBD Network

Each subnetwork can get another branching, thus multiple branches and thus a widely ramified construction of *sub-networks* is possible within the main network.

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor.

7.6.1.28. Update Parameters

Default Shortcut: <CTRL>+<U>

This command can be used in FBD or LD editor to update the parameters (inputs, outputs) of a box, which is already inserted in a network, after having changed its interface for example by adding an output.

The already defined connections of inputs and outputs remain unchanged, resp. if an input or output gets added, this will get the *???* and can be assigned.

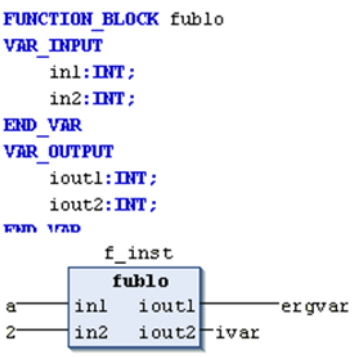

Figure 139: Original Version of Function Block with two outputs

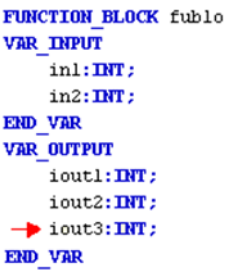

Figure 140: Function block modified, added output.

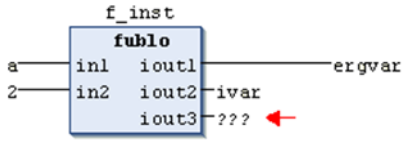

Figure 141: Updated function block by *Updated Parameters* in the FBD network

7.6.1.29. Remove Unused FB Call Parameters

Symbol:

This command is available only for implementation in FBD. Its execution will remove all unassigned entries or exits of the FB box focused, that is, all entries or exits whose assignments are empty or marked by *???*. However, the minimum number of necessary in- or outputs of the box will be maintained.

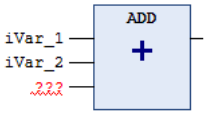

Figure 142: Remove Unused Parameters in FBs

7.6.1.30. View as function block diagram (FBD)

Default Shortcut: <CTRL>+<1>

This command is available if you currently are working in the Ladder Diagram (LD). It can be used in offline and online mode.

The LD networks will be converted appropriately to FBD networks. Notice anyway, that there are some special elements, which cannot get converted and thus will only be available in the appropriate editor view. Further on some constructs might not be converted un-ambiguously.

To switch back to the LD view use command *View as ladder logic*.

> **ATTENTION**
>
> A proper conversion presumes syntactically correct code. Otherwise, parts of the implementation can get lost.

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor.

7.6.1.31. View as ladder logic (LD)

Default Shortcut: <CTRL>+<2>

This command is available, if you currently are working in the Function Block Diagram (FBD) or Instruction List (IL) view of the POU. It can be used in offline and online mode.

Notice anyway, that there are some special elements, which cannot get converted and thus will only be available in the appropriate editor view. Further on some constructs might not be converted un-ambiguously.

FBD elements, which cannot be displayed as LD elements (for example XOR), will be displayed as FBD boxes within a LD network.

> **ATTENTION**
>
> A proper conversion presumes syntactically correct code. Otherwise, parts of the implementation can get lost.

Note: Concerning the view options for the components of a FBD/LD network notice the settings in the FBD, LD and IL Editor.

7.6.2. CFC Commands

The commands of this category are available for programming in the CFC editor.

Available commands:

- Select All
- Negate
- EN/ENO
- Set/Reset
  - None
  - R (Reset)
  - S (Set)
  - REF= (Reference Assignment)
- Execution Order
  - Display Execution Order
  - Set Start of Feedback
  - Send to Front
  - Send to Back
  - Move Up
  - Move Down
  - Move Down
  - Order by Data Flow

- Order by Topology
- Set Execution Order..

- Pins
  - Use Attributed Member as Input
  - Reset Pins
  - Remove Unused Pins
  - Add Input Pin
  - Add Output Pin
- Routing
  - Route All Connections
  - Create Control Point
  - Remove Control Point
  - Unlock Control Point
- Group
  - Create Group
  - Ungroup
- Edit Parameters..
- Connect Selected Pins
- Edit Worksheet
- Expand All Inline Monitoring Fields
- Collapse All Inline Monitoring Fields

7.6.2.1. Select All

Default Shortcut: <CTRL>+<SHIFT>+<A>

The command select all existing elements in the open editor.

7.6.2.2. Negate

Symbol:

Default Shortcut: <CTRL>+<N>

This command is used to negate inputs, outputs, jumps or RETURN commands. The symbol for the negation is a small circle on the connection. To assign a negation select the respective input or output pins of the element and perform the command. See **CFC Editor, Cursor Positions** for the possible cursor positions for selection (IEC 61131 Programming Manual).

A negation can be deleted by negating again.

7.6.2.3. EN/ENO

Symbol:

This command is used to give a selected block (Cursor position 3) an additional Boolean enable input EN and a Boolean output ENO (Enable Out).

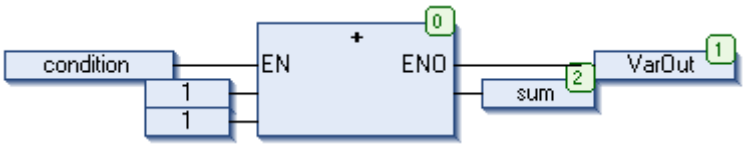

Figure 143: Add-box with EN/ENO

In this example, ADD will only be executed if the boolean variable condition is TRUE. VarOut will be set to TRUE after the execution of ADD. Notice that if afterwards condition changes to FALSE, ADD will not be executed any longer and also VarOut will be set to FALSE.

The example in figure below shows how the ENO value can be used for further blocks.

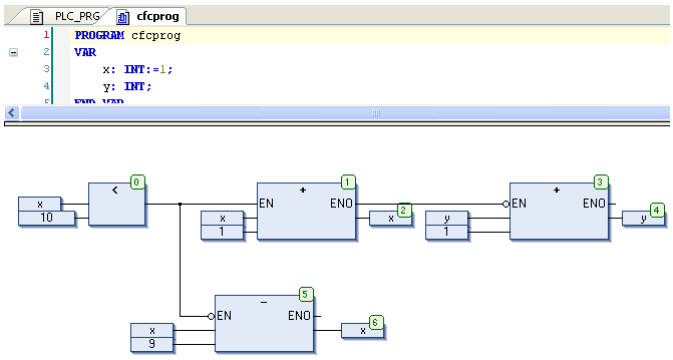

Figure 144: Use of EN/ENO

The numbers in the right corner of the boxes indicate the order in which the commands are executed.

For this example initialize x with *1*.As long as x is less than 10 (0), it will be increased by one (1). As soon as x = 10 the output of LT(0) will deliver the value FALSE and SUB (6) and ADD (4) will be executed. x will be set back to the value 1 and y will get increased by 1. LT (0) will be executed again as long as x is less than 10. Thus y is counting how often x passes though the value range 1 to 10.

7.6.2.4. Set/Reset

7.6.2.4.1. None

Symbol:

This command, per default part of the submenu *Set/Reset* of the CFC menu, removes a *Set* or *Reset* from an output element. Select the input pin of the respective output and perform the command. The *S* or *R* symbol at the output element will disappear.

See **CFC Editor, Cursor Positions** for the possible cursor position for selection (IEC 61131 Programming Manual).

7.6.2.4.2. R (Reset)

Symbol:

This command, by default part of the submenu Set/Reset of the CFCmenu, assigns a *Reset* to a boolean output element, which means that the output will be reset by a TRUE of the input and keep this value even if the input becomes FALSE again.

Select the input pin of the respective output and perform the command. See **CFC Editor, Cursor Positions** for the possible cursor position for selection (IEC 61131 Programming Manual).

The reset output will be indicated by a *R*.

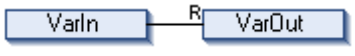

Figure 145: Reset

In this example VarOut will be set to FALSE, if VarIn delivers TRUE. VarOut retains this value, even when VarIn springs back to FALSE.

Alternative settings concerning the Set/Reset properties of an output are *None*, which means that there is no Set or Reset activity assigned, or S (Set).

7.6.2.4.3. S (Set)

Symbol: 

This command, by default part of the submenu *Set/Reset* of the CFC menu, assigns a *Set* to a boolean output element, which means that the output gets set by a TRUE of the input and keeps this value even if the input becomes FALSE again.

Select the input pin of the respective output and perform the command. See *CFC Editor, Cursor Positions* for the possible cursor position for selection (IEC 61131 Programming Manual).

In this example VarOut will be set to TRUE, if VarIn delivers TRUE. VarOut retains this value even when VarIn changes back to FALSE.

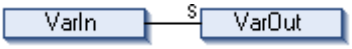

Figure 146: Set

Alternative settings concerning the *Set/Reset* properties of an output are *None*, which means that there is no *Set* or *Reset* activity assigned, or R (Reset).

7.6.2.4.4. REF= (Reference Assignment)

Symbol: 

Default Shortcut: <Ctrl>+<M>

The command assigns a reference to an Output element. It requires the CFC editor to be activated and an Output element input to be selected.

```
ref_int : REFERENCE TO INT;
a : INT;
```

Figure 147: Reference Assignment

7.6.2.5. Execution Order

In text-based and network-based editors, the execution order in POUs is uniquely defined. However, in the CFC editor, elements can be positioned freely, making the initial execution order non-unique. To resolve this, MasterTool IEC XE determines execution order based on data flow and the topological position of elements across multiple networks, sorting them from top to bottom and left to right. This ensures a unique, time and cycle-optimized processing order.

You can temporarily display the current execution order in the chart. When programming feedback networks, you can designate a starting point in the loop.

To explicitly edit the execution order in a CFC object, set the *Auto Data Flow Mode* property to *Explicit Execution Order Mode*. This allows you to adjust the execution order using menu commands.

7.6.2.5.1. Display Execution Order

Symbol: 

This command needs *Auto Data Flow Mode* to be visible.

The numbers indicate the automatically determined execution order, based on data flow. For multiple networks, the order depends on their topological position in the editor. Tags are hidden when you click in the CFC editor.

7.6.2.5.2. Set Start of Feedback

Symbol: 

This command needs *Auto Data Flow Mode* to be visible.

In the CFC editor, the starting point in feedback loops is marked with the  symbol and given the lowest execution order number. At runtime, feedback processing starts with this element.

7.6.2.5.3. Send to Front

Symbol: 

This command is part of submenu *Execution Order* in the CFC menu effects that all selected elements will be moved to the front of the execution order. Thereby the order within the group of selected as well as of the unselected elements is maintained. In addition, the order within the not selected elements will not be changed. Needs the *Explicit Execution Order Mode* to be visible.

7.6.2.5.4. Send to Back

Symbol: 

This command is part of submenu *Execution Order* in the CFC menu, effects that all selected elements will be moved to the end of the execution order. Thereby the order within the group of selected as well as of the unselected elements is maintained. In addition, the order within the not selected elements will not be changed. Needs the *Explicit Execution Order Mode* to be visible.

7.6.2.5.5. Move Up

Symbol: 

The Move Up command effects that all selected elements - with the exception of the element, which is at the beginning of the execution order - are moved one place forwards in the internal processing list.

The command is part of submenu *Execution Order* in the CFC menu. Needs the *Explicit Execution Order Mode* to be visible.

7.6.2.5.6. Move Down

Symbol: 

The *Move Down* command effects that all selected elements - with the exception of the element, which is at the beginning of the execution order - are moved one place backwards in the internal processing list.

The command is part of submenu *Execution Order* in the CFC menu. Needs the *Explicit Execution Order Mode* to be visible.

7.6.2.5.7. Order by Data Flow

This command is part of submenu *Execution Order* in the CFC menu. Needs the *Explicit Execution Order Mode* to be visible.

It effects that the execution order (indicated by the element numbers in the upper right corner of an element) in the CFC-Editor gets determined by the data flow of (all) elements and not by their position (topology).

The advantage of the order according to data flow is that an output box, which is connected to the output pin of a block, immediately will be processed after the block, which is not always so in case of a topological process flow. A topological order of processing might deliver another result in some cases than a processing by data flow. This can be recognized from the above-described example.

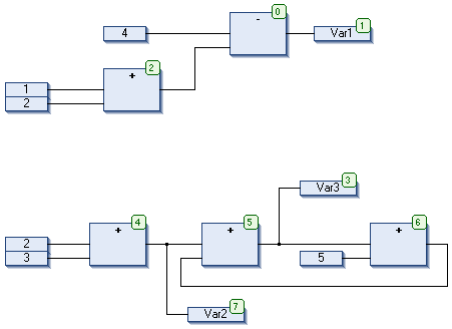

Figure 148: Order by Topology

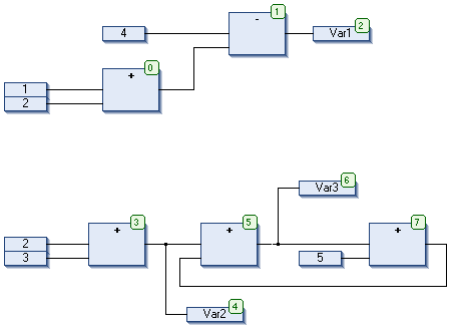

Figure 149: Processing According to Data Flow

When the command gets executed, the following will happen internally: First, the elements are ordered topographically. Then a new sequential processing list will be created. Based on the known values of the inputs, the computer calculates which of the not yet numbered elements can be processed next. In the above shown *network*, e.g. the ADD block (0) could be processed immediately since the values at its inputs (*1* and *2*) are known. Block SUB (1) can only be processed afterwards since the result from ADD must be known first, etc. Feedback paths get inserted last. Therefore, a sequencing by data flow will result.

7.6.2.5.8. Order by Topology

This command is part of submenu *Execution Order* in the CFC menu. It effects that the execution order in the CFC-Editor is determined by the topological order of the elements and not by the data flow.

Topological order means that the execution order, that is the processing of the elements runs from left to right and from top to bottom. The element numbers indicating the position of an element within the processing list, increase from left to right and from top to bottom. The position of the connection lines is not relevant, only the location of the elements is important.

When the command is executed, implicitly all currently selected elements get removed from the processing list and then re-inserted one by one in the remaining list from bottom right through to upper left. In doing so each selected element will be entered before its topological successor and the numbers of the remaining elements will be adapted.

Example, Topological arranging of selected elements:

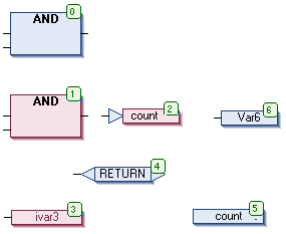

Figure 150: Sequence Before

The elements with numbers 1, 2 and 3 are selected. If now the command Order By Topology is executed, the elements first will be taken out of the sequential processing list. The subsequent re-inserting will be done conversely:

First ivar will be inserted ahead of label count, thus getting number 4, which makes RETURN fall back to 3.

Then jump count gets inserted ahead of Var6 and thus gets number 5. This effects that label count (before then having 5), output ivar3 and RETURN each get numbered down by 1.

At last the AND box will be re-inserted ahead of jump count and thus will get number 4. This again effects a reducing of the numbers each for label count (before then having 4), output ivar3 and RETURN by 1. Therefore, the following new order of execution will arise:

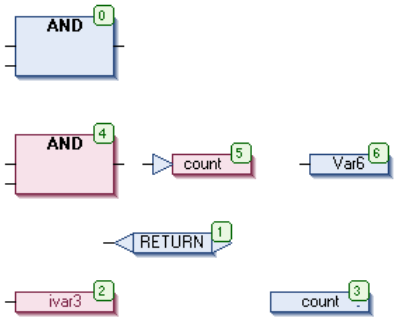

Figure 151: Sequence Afterwards

Also, a new element always will be inserted in the sequential processing list ahead of its topological successor.

7.6.2.5.9. Set Execution Order...

This command is part of submenu Execution Order in the CFC menu. It serves to redefine the element number of the currently selected element in order to change the position of this element within the execution order.

The command opens the dialog Set Execution Order. The current element number is displayed in field Current Execution Order and you can enter the desired new one in New Execution Order. The possible values are displayed in brackets.

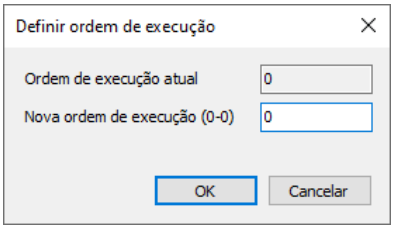

Figure 152: Set Execution Order

7.6.2.6. Pins

7.6.2.6.1. Use Attributed Member as Input

Symbol: 

In a CFC editor, when a function block input is selected, it is possible to connect a structure member to a scalar type input using the *Pins* menu or the *Pins* option in the context menu. To successfully connect the structure member to the input of the subsequent function block, the member must be provided with the *{attribute 'ProcessValue'}* pragma. Additionally, the data type of the structure member must be compatible with the data type of the subsequent input. When inputs are connected in this manner, they are marked with the *V* symbol. If the *Use Attributed Member as Input* command is not executed for this connection, a compiler error will be issued.

```
TYPE QINT :
STRUCT
```

```
    Status : STRING;
    {attribute 'ProcessValue'}
    Value1 : INT;
    Value2 : INT;
  END_STRUCT
  END_TYPE
```

```
  PROGRAM PLC_PRG
  VAR
    input1: QINT;
    output1: QINT;
    intValue: INT;
  END_VAR
```

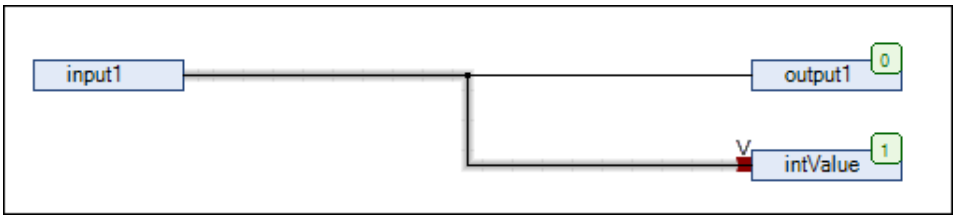

Figure 153: Use Attributed Member as Input

7.6.2.6.2. *Reset Pins*

Symbol:

If unused input or output pins have been removed from a box in the CFC Editor, for example because they are not used, or if the interface of the POU, which is represented by the box, has been changed, this command can be used to restore and update the display of the pins. It can also be used to display the parameters of type VAR_IN_OUT of a function block, which are hidden per default.

In the following example, input fbin2 of a function block instance had been deleted because it is not used. By selecting the fb1 box and using command restore pins all inputs and outputs of the function block, as defined in its implementation, can be displayed again.

That was the input and output declarations:

```
FUNCTION_BLOCK CFC_1
VAR_INPUT
    fbin1: INT;
    fbin2: INT;
END_VAR
VAR_OUTPUT
    fbout1: INT;
END_VAR
```

Figure 154: Reset Pins Declarations

The box ended up that way:

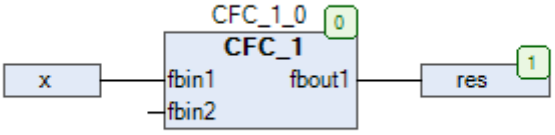

Figure 155: Before Reset Pins command being executed

And after the input fbin2 being removed and the command being executed, the box ended up that way:

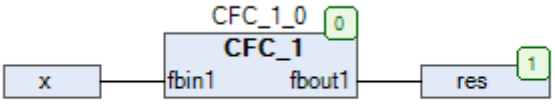

Figure 156: Before Reset Pins command after executed

7.6.2.6.3. Remove Unused Pins

This command removes non-connected pins from program, function block or non-local action calls in the current editor selection. This will not be done for function, method or operator calls as this would lead to invalid syntax.

7.6.2.6.4. Add Input Pin

Symbol: 

The command adds an additional input to the selected box. To use it, go to the CFC editor, open the Pins menu, or select Pins from the context menu. This function requires that a CFC editor is active and a box is currently selected.

7.6.2.6.5. Add Output Pin

Symbol: 

The command adds an additional output to the selected box. To use this function, ensure that a CFC editor is active and that a suitable box is selected.

7.6.2.7. Routing

The *Routing* submenu of commands is used to manage and organize the flow of signals and data paths between different components or elements within the CFC editor.

7.6.2.7.1. Route All Connections

Symbol: 

The command undoes all manual changes to the connections in the program, restoring them to their original state. However, connections fixed with control points cannot be automatically routed, so you must remove these control points before executing the command by clicking on *Remove Control Point*. Additionally, for manually changed connections marked with the icon, you will need to click *Unlock Connection* to proceed.

7.6.2.7.2. Create Control Point

Symbol: 

The command creates a control point at a connector. To use it, position the cursor over a connection. When you execute the command, the control point will be placed at the location on the connecting line where the cursor is positioned.

7.6.2.7.3. Remove Control Point

To use the command, ensure you have selected a connecting line. When you hover the mouse pointer over the selected line, existing control points will be displayed with yellow circle icons.

Position the cursor over the control point you want to delete and execute the command.

7.6.2.7.4. Unlock Control Point

Symbol: 

This command unlocks a locked connection. To use it, select a connection or connection mark. A connection becomes locked when you manually alter it during automatic routing. To perform automatic routing again, you must first unlock any locked connections.

You can also disconnect a locked connection by clicking the  icon associated with it.

7.6.2.8. Group

The *Group* submenu of commands is used to organize and manage multiple selected elements by grouping them together, allowing for simultaneous movement and manipulation while maintaining their individual positions.

7.6.2.8.1. Create Group

Symbol: 

The command groups the selected elements, which must be multiple elements. Once grouped, the elements can only be moved together, though their individual positions are not affected by the grouping. To ungroup the elements, use the Ungroup command.

7.6.2.8.2. Ungroup

Symbol: 

The command removes a previously created grouping. To use it, select the grouping you want to remove.

7.6.2.9. Edit Parameters...

The *Edit Parameters...* command opens a dialog to manage the constant input parameters of a function block.

The instantiated function block must have a *VAR_INPUT CONSTANT* to the command be enabled. In this case, the *Parameter* word will be visible in the lower left corner of the box, and the dialog can be opened by there.

Note: Only the CFC editor supports this functionality for *VAR_INPUT CONSTANT* variables. In contrast, the FBD editor in MasterTool IEC XE always displays all input parameters on the box, regardless of whether they are declared as *VAR_INPUT* or *VAR_INPUT CONSTANT*. MasterTool IEC XE also does not differentiate between these in text editors.

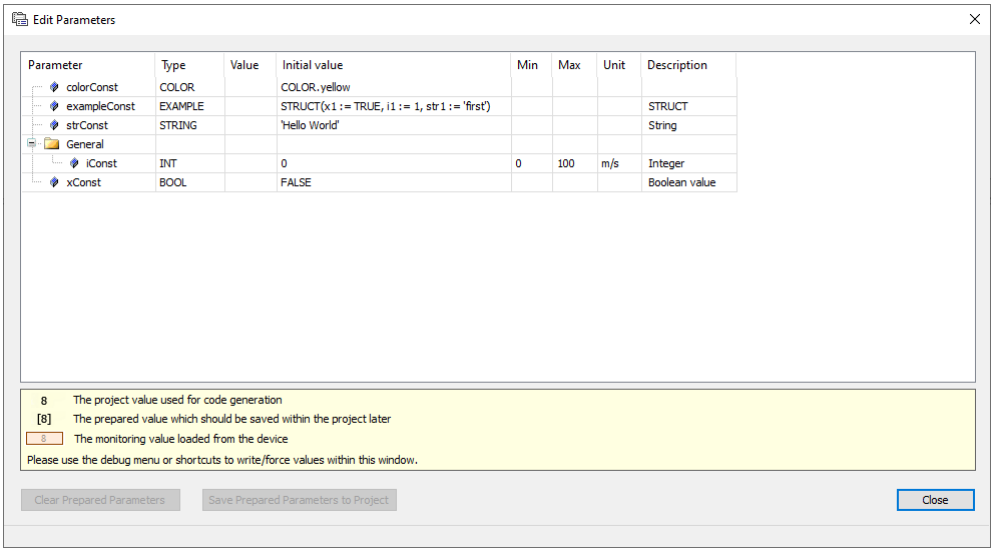

Figure 157: Edit Parameters

7.6.2.10. Connect Selected Pins

Symbol:

This command is only enabled when exactly one output pin and one or several input pins are selected. When executed, a connection between the output pin and the input pin(s) is established.

7.6.2.11. Edit Worksheet

This command opens the *Edit Worksheet* dialog for modifying the size of the working area of the current CFC.

The size of the work sheet is defined by the height and width of a rectangular area having its origin (X: 0, Y: 0) in the upper left corner of the editor window and including all existing CFC elements. Height and width are specified in number of grid units whereby the size of a grid unit is not changeable by the user. Height (Y): increasing positive values from top to down, width (X): increasing positive values from left to right.

The maximum size is 2048 grid units in width and height.

- *Use the following dimensions: If this option is activated, the size of the worksheet will be determined by the following width and height values*:
  - *Width*: Shows the current width in grid units. Can be edited, whereby it will not be possible to enter a width smaller than that, which is actually required by the existing elements. Increasing the value (X) will enlarge the width horizontally to the right.
  - *Height*: Shows the current height in grid units. Can be edited, whereby it will not be possible to enter a height smaller than that, which is actually required by the existing elements. Enlarging the value (Y) will enlarge the working sheet vertically downwards.
- *Adapt the dimensions automatically*: This option is activated by default. The size of the working sheet is defined by the bottommost (height) and the rightmost (width) element borders within the editor window. The origin (X=0, Y=0) is in the upper left corner.

The shift might not lead to an upper left corner less than 0/0. If option Use the following dimensions is activated in the upper part of the dialog, the shift may not exceed the width and height defined there. If option Adapt the dimensions automatically' is activated, the shift might exceed the current dimensions and the width and height values will be updated accordingly.

- *Move the working sheet origin relatively*: If this option is activated, the working sheet can be shifted vertically and/or horizontally by the offset values given in the following.
  - *X offset*: By default is 0. Entering a positive value shifts the chart to the right, thus possibly increasing the width of the working sheet. Entering a negative value shifts the chart to the left, thus only possible, if there is space between the leftmost element and the left window border.
  - *Y offset*: By default is 0. Entering a positive value shifts the chart downwards, thus possibly increasing the height of the working sheet. Entering a negative value shifts the chart upwards, thus only possible, if there is space between the uppermost element and the upper window border.

If you enter invalid sheet size values, an error message dialog will pop up, also listing the given restrictions.

7.6.2.12. Expand All Inline Monitoring Fields

The command opens all collapsed inline monitoring fields, which are marked with the > or < symbols. To expand a single inline monitoring field, click the > or < symbol displayed at its location.

Note: The CFC editor must be in online mode, and the implementation part must be in focus.

7.6.2.13. Collapse All Inline Monitoring Fields

The command closes all expanded inline monitoring fields, which will then be marked with the > or < symbols. To collapse a single inline monitoring field, hover over it and click the *x* symbol.

Note: The CFC editor must be in online mode and the implementation part is in focus.

7.6.3. SFC Commands

The commands category SFC are available for programming in the SFC editor:

- Init Step
- Add Entry Action
- Add Exit Action
- Insert Step Transition
- Insert Step Transition After
- Parallel
- Alternative
- Insert Branch
- Insert Branch Right
- Insert Action Association
- Insert Action Association After
- Insert Jump
- Insert Jump After
- Insert Macro
- Insert Macro After
- Zoom Into Macro
- Zoom Out of Macro
- Paste After
- Toggle step between active/inactive

7.6.3.1. Init Step

This command is used in the SFC editor to transform the currently selected step to an init step.

Thus, the frame of the step element will change to a double-line. The step previously having been the init step will automatically change to a normal step and get displayed now with a single line frame. This transformation might be useful if you want to reconstruct an existing chart.

When creating a new SFC POU automatically an init step element will be inserted followed by a transition (TRUE) and a jump back to the init step.

Notice the possibility to set back the SFC to the init step by using variables SFCInit and SFCReset.

7.6.3.2. Add Entry Action

Symbol: 

When you use this command, it opens the *Add Entry Action* dialog, allowing you to define a new entry action. Depending on the SFC options, a dialog prompt might appear to let you select the duplication mode for the new step action. For this to work, a step element in SFC must be selected. Once the entry action is defined, it will automatically open in the ST editor, and the step element will display an *E* in its lower-left corner, indicating the presence of the entry action.

7.6.3.3. Add Exit Action

Symbol:

Using this command opens the *Add Exit Action* dialog, where you can define a new exit action. Depending on the SFC options, a dialog prompt may appear to select the duplication mode for the new step action. For more information, refer to the help page for the *Insert Entry Action* command. It is important to note that a step element in SFC must be selected for this process to work.

7.6.3.4. Insert Step Transition

Symbol:

This command is used in the SFC-Editor to insert a step and a transition before the currently selected position.

The positioning (sequence) of the new step and transition depends on whether a step or transition have been selected when performing the insert command. Automatically the sequence step-transition-step-transition-... will be kept. See the figures below for examples.

Step 0 is selected and *Insert Step Transition* is executed:

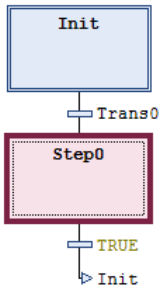

Figure 158: Before command being execute

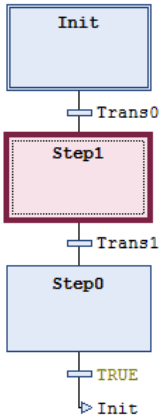

Figure 159: After command being execute

TRUE transition is selected and *Insert Step Transition* is executed:

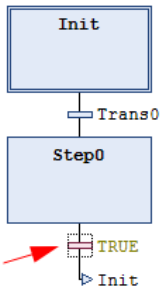

Figure 160: True Transition before command execution

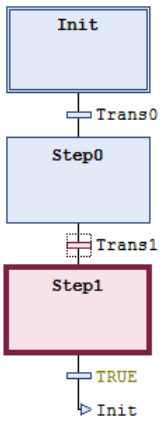

Figure 161: True Transition after command execution

7.6.3.5. Insert Step Transition After

Symbol:

This command is used in the SFC-Editor to insert a step and a transition after the currently selected step or transition.

The positioning (sequence) of the new step and transition depends on whether a step or transition have been selected when performing the insert command. Automatically the sequence step-transition-step-transition... will be kept.

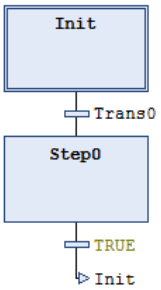

Figure 162: Step and Transition Inserted

In this example, the new step and transition are placed after transition TRUE, which had been selected when performing the *Insert* command.

The new step by default is named *Step<n>*, whereby n is a running number starting with *0* for the first step which is inserted in addition to the init step.

The new transition correspondingly by default is named *Trans<n>*.

To modify the default names perform a mouse-click on the name string in order to get into the edit mode.

7.6.3.6. Parallel

Symbol:

This command available in the SFC Editor transforms the currently selected alternative branch to a parallel branch.

Notice that after a branch transformation you must check and adapt the chart appropriately, that is you must arrange steps and transitions as required for the respective type of branching.

7.6.3.7. Alternative

Symbol:

This command available in the SFC Editor transforms a parallel branch to an alternative branch.

Notice that after a branch transformation you must check and adapt the chart appropriately, that is you must arrange steps and transitions as required for the respective type of branching.

7.6.3.8. Insert Branch

Symbol:

This command is used in the SFC-Editor to insert a branch left to the currently selected element(s).

7.6.3.9. Insert Branch Right

Symbol:

This command is used in the SFC-Editor to insert a branch right to the currently selected element(s). To insert it each left to the currently selected step, use command *Insert branch*.

- If the uppermost element of the current selection is a transition or an alternative branch, an alternative branch will be created.
- If the uppermost element of the current selection is a step, a macro, a jump or a parallel branch, a parallel branch with label *Branch<x>* will be inserted. This is a default label name where x is a running number. You can edit the label name. The branch label might be used as a jump target.
- If currently a common element of an existing branch is selected (horizontal line), the new branch will be added to the existing branches on the right most position. If currently a complete arm of an existing branch is selected (horizontal line), the new branch will be added directly right to that one.

Note: Notice that branches can be transformed by commands *Alternative* and *Parallel*.

7.6.3.9.1. Example of Parallel Branch

In see a new parallel branch, created by command *Insert Branch Right* when *Step 1* was selected. Automatically a step (*Step 2* in the example) gets inserted.

Processing in online mode: When *Trans0* is TRUE, *Step 2* will be executed immediately after *Step 1* before *Trans 1* is noticed. So both branches will be executed, in contrast to alternative branches.

7.6.3.9.2. Example of Alternative Branch

In the figure below see a new alternative branch, created by command *Insert Branch Right* when transition *Trans1* was selected. Automatically a step *Step 2* and a preceding and a subsequent transition (*Trans2*, *Trans3*) get inserted.

Processing in online mode: When *Step 0* is active, the following transitions (*Trans1*, *Trans2*) will be checked from left to right. The first branch whose transition is found to be TRUE, will be executed. Thus, only one branch is executed, in contrast to parallel branches.

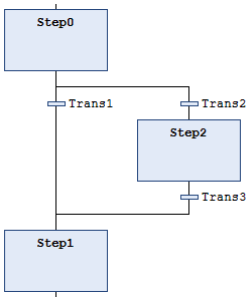

Figure 163: Alternative Branch

7.6.3.10. Insert Action Association

Symbol:

This command is used in the SFC-Editor to associate an action to a step.

Select the desired step and perform the command. The action box will be inserted right to the step box.

If already one or several action(s) are associated to a step, the new action element will be placed - as first action (upper position) for the step, if the step has been selected when performing the Insert command- directly before the action, which was selected when performing the insert command.

See also Insert Action Association After.

The left part of an action box contains the action qualifier, by default *N*, in the right part an action name must be entered. For this purpose click on the field to open an edit frame. The action must be available in the project.

The qualifier also can be edited inline. For valid qualifiers see the corresponding item.

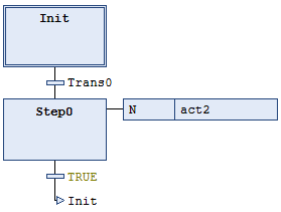

Figure 164: Act2 Associated with Step0

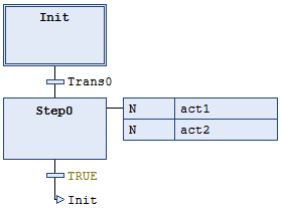

Figure 165: Action Association to Step0

7.6.3.11. Insert Action Association After

Symbol:

This command is used in the SFC-Editor to associate a further action to a step after an existing one.

Select the desired step and perform the command. The action box will be inserted right to the step box.

If already one or several action(s) are associated to a step, the new action element will be placed - as last action (lowest position) associated to the step, if the step was selected when performing the Insert command- directly after the action, which was selected when performing the Insert command.

The left part of an action box contains the action qualifier, by default *N*, in the right part an action name must be entered. For this purpose click on the field to open an edit frame. The action must be available in the project.

The qualifier also can be edited inline. For valid qualifiers see the corresponding item.

7.6.3.12. Insert Jump

Symbol:

This command is used in the SFC-Editor to insert a jump element before the currently selected element.

The new jump automatically is provided with *Step* specifying the target of the jump. Replace this string by the name of a step or by the label of a parallel branch, which should be jumped to.

7.6.3.13. Insert Jump After

Symbol:

This command is used in the SFC-Editor to insert a jump element after the currently selected element.

Jumps may only be used at the end of an alternative branch.

The new jump automatically is provided with *Step* specifying the target of the jump. Replace this string by the name of a step or by the label of a parallel branch, which should be jumped to.

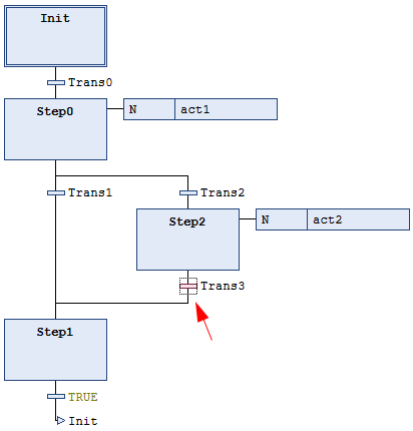

Figure 166: *Insert Jump After* command before being executed

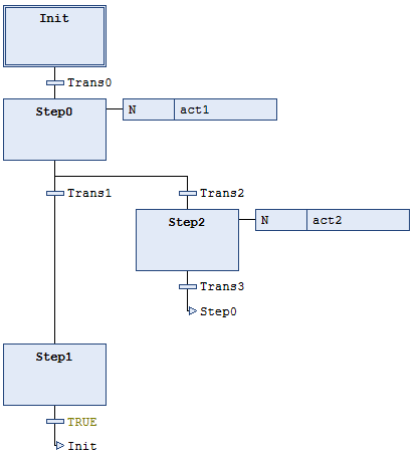

Figure 167: *Insert Jump After* command after being executed

7.6.3.14. Insert Macro

Symbol:

This command is used in the SFC-Editor to insert a macro box before the currently selected position in the diagram. By default, the macro name *Macro<x>* will be entered in the box, whereby x is a running number. You can edit the macro name.

To edit or view a macro the macro editor can be opened via command *Zoom Into Macro*.

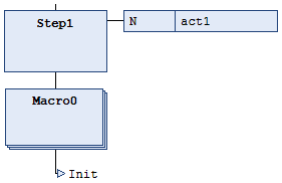

Figure 168: Macro Selected in SFC Diagram

7.6.3.15. Insert Macro After

Symbol:

This command is used in the SFC editor to insert a macro after the currently selected position in the diagram.

7.6.3.16. Zoom Into Macro

Symbol:

This command is used in the SFC-Editor to zoom into a macro that is to open the macro editor view.

The command can be used in offline and online mode.

Select the macro box in the SFC diagram and perform the command. The main SFC editor view will disappear and instead the macro editor will be opened. Here you can edit or just view the section of the chart which is just represented by the macro box in the main SFC view. The zoom menu as usual for editor views is available in the lower right corner.

To return to the SFC standard view by command *Zoom Out of Macro*.

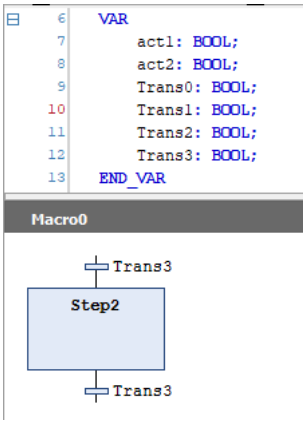

Figure 169: Macro Editor

7.6.3.17. Zoom Out of Macro

Symbol:

This command is used in the SFC-Editor to close the macro editor, which is currently opened in order to return to the main SFC editor view. The command can be used in offline and online mode.

7.6.3.18. Paste After

Symbol:

The command inserts the elements from the clipboard after the selected position.

7.6.3.19. Toggle step between active/inactive

Symbol:

The command toggles a step between active and inactive while in online mode. This functionality requires the application to be in online mode. When a step is selected, you can use this command to switch the step's status between active and inactive.

7.7. Menu Text List

The *Textlist* menu provides commands for editing a a text list. Per default these commands are available in a *Textlist* menu and in the context menu when working in a text list, and the appropriate ones also in the *Visualization* menu.

Available commands:

- Insert Text
- Add Language
- Remove Language
- Rename Language
- Import/Export Textlists
- Export All .txt Text List Files
- Export All Unicode .txt Text List Files
- Update Visualization Text IDs
- Check Visualization Text IDs
- Remove Visualization Text IDs
- Add text list support
- Remove text list support

7.7.1. Insert Text

Symbol:

This command of category *Text list* is available in a *Textlist* menu and in the context menu when working in the Text List editor. It serves for adding a new text in a new line above that one where currently the focus is on. It opens an edit field in the *Standard* column and by default enters the string *NewText*, which you can modify immediately.

7.7.2. Create Global Text List

Symbol:

The *GlobalTextList* global text list is created in the POUs view when a visualization is open.

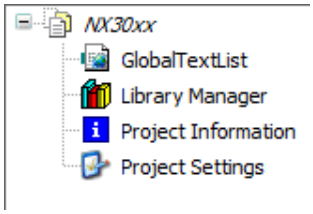

Figure 170: *GlobalTextList* object

7.7.3. Add Language

Symbol: 

This command of category *Text list* is used to add a new language column to a textlist.

Open the textlist, perform the command and enter the language name in the *Choose Language* dialog. After confirming with *OK* the language column will be added rightmost in the current list.

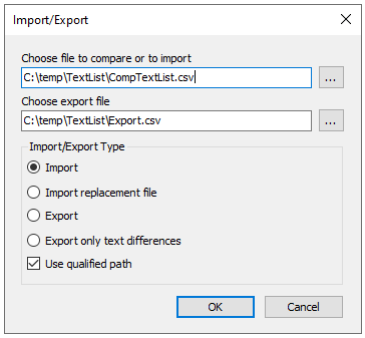

Figure 171: Dialog Import/Export Textlists

7.7.4. Remove Language

Symbol: 

This command of category *Text list* is used to remove that language column from a textlist. Open the textlist, set the cursor in the respective column and perform the command.

7.7.5. Rename Language

Symbol: 

This command of category *Text list* is used to rename that language column.

7.7.6. Import/Export Textlists

Symbol: 

This command of category *Text list* provides the data exchange with other programs as for instance Excel. The data format in use is .csv (Comma Separated Values). The executing the command the following dialog appears:

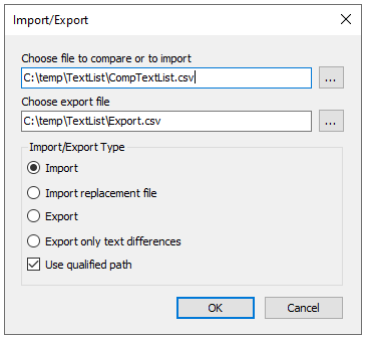

Figure 172: Dialog Import/Export Textlists

By entering the corresponding paths or by making use of the input assistant (...) the files to be imported, exported or compared may be specified. Which of the actions should be executed, may be defined by activating the corresponding item in the lower part of the dialog:

- *Import*: Importing an external file its dataset is put in line with the dataset of the project. The dataset in the project is adjusted according to the following rules:
  - If the data content is identical, the data set will be left unchanged.
  - If a translation has been added to the external file, it will be added to the data set of the project as well.
  - If text within a translation has been modified, the modification will be overtaken in the project as well.
  - If translation texts are missing within the external file, the data set in the project will not be modified.
  - If a new line has been added to the external file, this new data record will be incorporated in the data set of the project file.
  - If the project contains an additional line, it will be preserved.
  - A modification within the column Default may be considered as insertion of new text. If however there are text positions containing several empty spaces instead of a single one, then this will not be handled as a modification.
- *Import replacement file*: While importing a textlist a modification within column *Default* is considered as an insertion of a new line. The reason is, that the column *Default* is serving as key for comparing the lines during import/export. Note: If texts in the column have several blanks instead of one, this is not considered as a change.
- *Export*: The text lists of the selected project are exported, with all available project languages included as columns in the export file. This file can be used for the external translation of language-dependent texts. If the option to export one file per text list is enabled, each selected text list will be exported to a separate file. Otherwise, all selected text lists will be combined and exported together into a single file.
- *Export only text differences*: For the comparison process, an import file must be selected in *Choose file to compare or to import*, and an export file must be chosen in *Select export file*. The import file is then read, and the lines of the active text list are compared with it. Matching lines are ignored, while differing lines are written to the export file. If needed, translations from the text list are accepted, and translations from the import file are accepted and overwritten as necessary.
- *Use qualified path*: If this option is selected, text lists are exported and imported with their full, qualified path name. This ensures that text lists are uniquely identified, even if they appear multiple times in the project with the same name.

7.7.6.1. Example – Import of .csv File

Default old	Default new	Command
Cancel ?	Cancel	**REPLACE**
Do you want to register ?	Do you really want to register ?	**REPLACE_AND_REMOVE**
Do you really want to register ?!	Do you really want to register ?	**REPLACE_AND_REMOVE**

Table 20: File Import

The replacement file will be executed top down. Thus, the change history might be accounted for.

The command defines what to do with a text line. The only commands available yet is REPLACE. It will have the following effect:

Normally, the text entered in column *Default* will be replaced by the new text. In the example *Cancel ?* will be replaced by *Cancel* and *Do you want to register ?* by *Do you really want to register ?*. Simultaneously the texts of all visualization elements will be adjusted, i.e. the old text entries within the visualization elements will be replaced.

In the case that the new default text is already contained in the *Default* field of another row of the textlist, the row containing the entry to be replaced will be deleted completely. The visualization elements concerned receive the corresponding entries of the remaining row with the same default entry. In the example this will happen for the default entry *Do you really want to register ?!* that should be replaced by *Do you really want to register ?*. Due to the change history there will already exist a row with this default entry when the related REPLACE command shall be carried out. To avoid multiple occurrences of the key, the row containing the *Default old* text *Do you really want to register ?!* will be deleted completely from the textlist.

7.7.6.2. Example – Importing .csv File

Data content of external file:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		
TextList1	2	Light		

Table 21: Data of External File

Data content of textlist of project before import:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abortion
TextList1	1	Door	Tür	Door
TextList2	3	Seat	Sitz	Seat

Table 22: Text List before Import

During the import all differences are incorporated into the project. Thereby the two lists are adapted so that he following textlist will result in the project.

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door	Tür	Door
TextList1	2	Light		
TextList2	3	Seat	Sitz	Seat

Table 23: Text List Resultant

7.7.6.3. Example - Export of a .csv File

Data set of external file:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abort
TextList1	1	Door	Tür	Door
TextList2	2	Seat	Sitz	Seat

Table 24: Data of External File

Data content of textlists of project before export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		
TextList1	3	Light		
TextList2				

Table 25: Data before Export

During the export all differences are incorporated in the external file. Thereby the two lists are adapted so that the following external file will be created.

Data content of textlists of project after export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door	Tür	Door
TextList1	3	Light		
TextList2	2	Seat	Sitz	Seat

Table 26: Data before Export

7.7.6.4. Example - Export of Text Differences Only

Data content of textlists of project before export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abort
TextList1	1	Door	Tür	Door
TextList2	2	Seat	Sitz	Seat

Table 27: Data of External File

Data content of textlists of project before export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		
TextList1	3	Light		
TextList2				

Table 28: Data before Export

During the export all lines differing from the corresponding ones (line 2,3 and 5 of the actual list) are included in the export file.

Data content of external file after export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	3	Light		

Table 29: Data after Export

7.8. Export All .txt Text List Files

Symbol: 

Essentially, this command does the same as Export All Unicode .txt Text List Files, except it export all text, not only Unicode.

7.9. Export All Unicode .txt Text List Files

Symbol: 

The function exports all text lists of the project, provided that a text list or global text list is open and active. In this process, the visualization encodes the characters of the texts into Unicode, assuming that the *Use Unicode strings* option is selected in the *Visualization Manager*. Additionally, the application's *VISU_USEWSTRING* compiler statement must be set. This can be verified by selecting the *Properties* command in the context menu, then navigating to the Build tab, where *VISU_USEWSTRING* should be specified in the Compiler definitions input field.

Upon meeting these conditions, a simple text file in .txt format is created for each text list, with the name of the text list serving as the file name. The directory for exporting these files is defined in the *Project* > *Project Settings* > *Visualization* menu, specifically on the *General* tab under Text list files. This format is compatible with a controller, meaning the file can be copied to a controller and configured in the *Visualization Manager* to prevent the text lists from being transferred again when the application is downloaded.

7.9.1. Update Visualization Text IDs

Symbol: 

This command belongs to category *Text List*. If a static text is modified within a visualization element, the visualization and evetually the GlobalTextList as well must have write permission. If modifications will be done, though the write permission is missing, it can be, that the text-ids do no longer fit to the texts of a visualization element.

By use of the command *Update Visualization-Text-IDs* these errors can be corrected automatically. Therefore all affected visualizations and the GlobalTextList must have write permission.

7.9.2. Check Visualization Text IDs

Symbol: 

This command belongs to category Text List. If a static text is modified within a visualization element, the visualization and evetually the GlobalTextList as well must have write permission. If modifications will be done, though the write permission is missing, it can be, that the text-ids do no longer fit to the texts of a visualization element.

By use of the command *Check Visualization-Text-IDs* such errors can be detected in the visualizations.

7.9.3. Remove Visualization Text IDs

Symbol: 

This command of category *Text list* serves to delete texts being no longer used in a visualization element from the GlobalTextList.

7.10. Add text list support

Symbol: 

When a text list support is added to a selected DUT object of type *Enumeration* via the context menu of a standard DUT object of this type, it enables the localization of the enumeration component identifier and allows the symbolic component value to be displayed in a text display within a visualization.

7.11. Remove text list support

Symbol: 

The text list support is removed from the selected enumeration object using the context menu of an enumeration object that has text list support. This support enables the localization of the enumeration component identifier and allows the symbolic component value to be displayed in a text display within a visualization.

7.12. Menu Visualization

The *Visualization* menu provides commands for editing a display object in the view editor.

Available commands:

- Interface Editor
- Hotkeys Configuration
- Visualization Element List
- Activate Keyboard Usage
- Order
  - Bring to Front
  - Bring One to Front
  - Send to Back
  - Send One to Back
- Alignment
  - Align Left
  - Align Top
  - Align Right
  - Align Bottom
  - Align Vertical Center
  - Align Horizontal Center
  - Make Horizontal Spacing Equal
  - Increase Horizontal Spacing
  - Decrease Horizontal Spacing
  - Remove Horizontal Spacing
  - Make Vertical Spacing Equal
  - Increase Vertical Spacing
  - Decrease Vertical Spacing
  - Remove Vertical Spacing
  - Make Same Width
  - Make Same Height

    - Make Same Size
    - Size to Grid
- Assisted Positioning
    - No Assistance
    - Snaplines
    - Grid
- Group
- Ungroup
- Background
- Multiply Visualization Element
- Add Node
- Add Child Node
- Add Category Node
- Add Standard Property Nodes
- Move Node Up
- Move Node Down

7.12.1. Interface Editor

Symbol: 

Default Shortcut: <ALT>+<F6>

This command (category *Visual Commands*) opens the *Basics of Interface Editor* for defining frame parameters in a visualization which is intended to get referenced in a *Frame* element in another visualization.

The tab provides an editor for declaring interface variables, similar to the function block declaration editor, but without initialization.

7.12.2. Hotkeys Configuration

Symbol: 

This command (category *Visual Commands*) opens the *Hotkeys Configuration* for the current visualization. It will be displayed in a tabbed view in the upper part of the visualization editor.

There will be possible to configure an sequence of keys to execute an action.

7.12.3. Visualization Element List

Symbol: 

This command (category *Visual Commands*) opens the *Element List* of the current visualization. It will be displayed in a tabbed view in the upper part of the visualization editor. This view lists the visualization elements in the open visualization. Grouped elements are shown in a tree structure, maintaining their order within each group.

7.12.4. Activate Keyboard Usage

Symbol: 

This command (category *Visual Commands*) is available in the menu bar for an integrated visualization. It activates resp. deactivates the keyboard usage in online mode.

When the keyboard operation is activated, any inputs on elements and the selection of elements can be done by using certain keys. In this case other commands given via key shortcuts will not be executed as long as the visualization editor is active and in online mode.

7.12.5. Order

The sub-menu is used to determine the order of elements within layers, ensuring that elements in the back layer are hidden by those in the front. This is essential for visualizations where elements are positioned in layers, one behind the other.

7.12.5.1. Bring to Front

Symbol: 

This command (category *Visual Commands*) places the selected element in the absolute foreground of the visualization.

7.12.5.2. Bring One to Front

Symbol: 

This command (category *Visual Commands*) places the selected element one layer higher, that means nearer to the foreground of the visualization.

7.12.5.3. Send to Back

Symbol: 

This command (category *Visual Commands*) places the selected element in the absolute background of the visualization.

7.12.5.4. Send One to Back

Symbol: 

This command (category *Visual Commands*) places the selected elements one layer deeper that means nearer to the background of the visualization.

7.12.6. Alignment

The sub-menu is used to align visualization elements within the area of the visualization window.

7.12.6.1. Align Left

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the left line of that of those elements which is at the left-most position.

7.12.6.2. Align Top

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the upper line of those elements which is at the up-most position.

7.12.6.3. Align Right

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the right line of that of those elements which is at the right-most position.

7.12.6.4. Align Bottom

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the bottom line of that of those elements which are at the most bottom position.

7.12.6.5. Align Vertical Center

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the vertical center of all selected elements.

7.12.6.6. Align Horizontal Center

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the horizontal center of all selected elements.

7.12.6.7. Make Horizontal Spacing Equal

Symbol: 

The command aligns the selected visualization elements so that the leftmost and rightmost elements remain in their positions, while the elements in between are spaced evenly along a horizontal line. This requires the selection of three or more elements, where the first element is blue and the others are gray.

7.12.6.8. Increase Horizontal Spacing

Symbol: 

The command aligns the selected visualization elements so that the blue element remains in its position, while the other elements are arranged horizontally with increasing space between them. The spacing grows by one pixel for each element. This requires multiple elements to be selected.

7.12.6.9. Decrease Horizontal Spacing

Symbol: 

The command aligns the selected visualization elements so that the blue element remains in its position, while the other elements are arranged horizontally with reduced spacing between them. The distance decreases by one pixel for each element. This requires multiple elements to be selected.

7.12.6.10. Remove Horizontal Spacing

Symbol: 

The command aligns the selected visualization elements so that the blue element remains in its position, while the other elements are arranged horizontally with no space between them. This requires multiple elements to be selected.

7.12.6.11. Make Vertical Spacing Equal

Symbol: 

The command aligns the selected visualization elements so that the top and bottom elements remain in their positions, while the elements in between are spaced evenly along a vertical line. This requires the selection of three or more elements, with the first element being blue and the others gray.

7.12.6.12. Increase Vertical Spacing

Symbol: 

The command aligns the selected visualization elements so that the blue element remains in its position, while the other elements are arranged vertically with increasing space between them. The spacing expands by one pixel for each element. This requires multiple elements to be selected.

7.12.6.13. Decrease Vertical Spacing

Symbol: 

The command arranges the selected visualization elements vertically, ensuring the blue element stays in place while the other elements are positioned with increasing space between them. The spacing grows by one pixel for each element, and this operation requires multiple elements to be selected.

7.12.6.14. Remove Vertical Spacing

Symbol: 

The command arranges the selected visualization elements vertically, keeping the blue element in its position while placing the other elements directly adjacent to each other with no space in between. This requires multiple elements to be selected.

7.12.6.15. Make Same Width

Symbol: 

The command adjusts the width of the selected visualization elements to match that of the blue element. This requires multiple elements to be selected, with the first element being blue and the others gray.

7.12.6.16. Make Same Height

Symbol: 

The command adjusts the height of the selected visualization elements to match that of the blue element. This requires multiple elements to be selected, with the first element being blue and the others gray.

7.12.6.17. Make Same Size

Symbol: 

The command adjusts the size of the selected visualization elements to match that of the blue element. This requires multiple elements to be selected, with the first element being blue and the others gray.

7.12.6.18. Size to Grid

Symbol: 

The command aligns the selected visualization elements according to their size and position on the grid. It requires multiple elements to be selected and does not apply to lines or polygons.

7.12.7. Assisted Positioning

Symbol: 

This feature enables the free positioning of elements, allowing them to be inserted and moved with complete flexibility.

7.12.7.1. No Assistance

Symbol: 

This feature enables the free positioning of elements, allowing them to be inserted and moved with complete flexibility.

7.12.7.2. Snaplines

Symbol: 

This feature provides positioning assistance with snaplines. When inserting and moving elements, lines appear to indicate positions that align with adjacent elements, as well as lines that subdivide the visualization editor into a grid. This allows you to align elements with neighboring ones or arrange them in a fixed column and row layout.

7.12.7.3. Grid

Symbol: 

This feature enables positioning along the displayed grid. When the grid function is active, elements are automatically aligned with grid points as they are inserted and moved, ensuring they adhere to the grid.

7.12.8. Group

Symbol: 

This command (category *Visual Commands*) groups the currently selected visualization elements and displays the group as a single selected object. For multiple selection keep the <SHIFT> key pressed while clicking on the desired elements. Alternatively you might perform a mouse-click outside of an element in the editor window and - while keeping the mouse-button pressed - draw a rectangle around the desired elements. For resolving the group use command Ungroup.

The following picture shows the grouping and ungrouping of three rectangles:

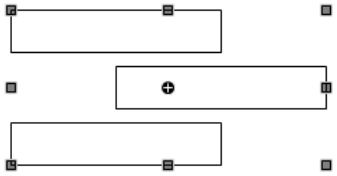

Figure 173: Grouping

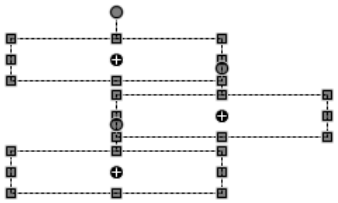

Figure 174: Ungrouping

7.12.9. Ungroup

Symbol: 

This command (category *Visual Commands*) resolves a selected group of elements. The particular elements will be displayed each selected. See also Group.

7.12.10. Background

Symbol: 

The command opens the *Background* dialog, where you can specify whether the background of the visualization is displayed as a color or an image.

- *Use color*: Present in *Color Settings*, if checked will allow to select a color.
- *Use image*: Present in *Image Settings*, if checked, this option allows referencing an image from the project's image collection, formally specified as the instance path <name of image pool>.<ID>.

7.13. Multiply Visualization Element

Symbol: 

The command opens the *Multiply Visualization Element* dialog, which provides a configuration based on the template element and the array declaration. In this dialog, you can rearrange the elements, adjust their quantity, and define the index access to the array data. Upon exiting the dialog, a field of similar elements is generated from the template element. The properties of these new elements are automatically updated with array variables, now configured with precise array indexes. These elements correspond to those where an array variable with index access placeholders was configured in the template. To use this command, the visualization must be active, and a configured template element must be selected.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Visualization/_visu_cmd_multiply_visualization_element.html.

7.14. Add Node

Symbol: 

This command adds a new property to the frame configuration in the tree view of the interface properties. The focus must be on the *Frame Configuration* tab. The new property is added to the bottom of the tree view, and you can double-click the entry to open the line editor and specify a suitable name. Additionally, you can use the *Move node up* command to reposition the property. To add another property under a specific category, select a property within that category and click *Add node*. This will insert a new property parallel to the selected one.

7.15. Add Child Node

Symbol: 

This command adds a new property to the frame configuration within the tree view of the interface properties, under the selected category. A category must be selected in the tree view. The new property is classified under this category, and you can double-click the entry to open the line editor to specify a suitable name. You can then use the *Move node up* command to adjust the property's position.

7.16. Add Category Node

Symbol: 

This command adds a new category to the frame configuration within the tree view of the interface properties. The focus must be on the *Frame Configuration* tab. The category is added to the bottom of the tree view, and you can double-click the entry to open the line editor and specify a suitable name. The *Move node up* command can then be used to adjust the position of the category.

7.17. Add Child Category Node

Symbol: 

This command adds a new category to the frame configuration in the tree view of the interface properties, under the selected category. A category must be selected in the tree view, and you can double-click the new entry to open the line editor and specify a suitable name.

7.18. Add Standard Property Nodes

Symbol: 

This command adds a standard property to the frame configuration in the tree view of the interface properties. The focus must be on the *Frame Configuration* tab. The property is added at the bottom of the tree view. When you double-click the new entry, a list box appears where you can select the desired standard property. These available properties are those of the frame element, which can be found in more detail on the *Frame* help page.

Once added, standard properties are disabled in the tree view because they cannot be modified.

Regarding input forwarding, a frame element that references a visualization template forwards input events to the referenced visualization, even after the frame has processed the input. The *Enter* and *Leave* events work for elements within the template visualization, allowing, for instance, hover effects to be implemented. This functionality is only available for visualization templates with frame configuration using a hierarchical editor.

7.19. Move Node Up

Symbol: 

This command moves the selected entry up by one position in the tree view. It requires an entry to be selected in the tree view for the action to be performed.

7.20. Move Node Down

Symbol: 

This function moves the selected entry down by one position in the tree view. It requires an entry to be selected in the tree view for the action to be executed.

7.21. Build Menu

Menu to help the user find errors and compile the writing code.

Available commands:

- Generate Code
- Build
- Clean
- Clean All

7.21.0.1. Generate Code

Symbol: 

Default Shortcut: <F11>

This command, allows compiling the currently active application just for test purposes. A code generation run will be done like by default is done when logging in with the application. However, no code will be downloaded and no compile info file will be created in the project directory. Thus, the user can check for any detected compilation errors before going online with a possibly incorrect code.

Note: Some changes made in the application are applied during the code generation process, such as changing the name of system tasks when they are changed the names of the corresponding instances. Similarly, some of project checks are performed only when this command is executed.

7.21.0.2. Build

The command initiates the compile operation for the active application. During this process, MasterTool IEC XE performs a syntactic check of all objects within the application, though it does not generate code as it would when logging into the target system or downloading the application. The build operation is automatically triggered whenever you log in with a modified program. Once the check is complete, any error messages or warnings are displayed in the message view under the *Build* category.

7.21.0.3. Clean

This command, deletes the compile information for the currently active application. The compile information was created and stored in a file **.compileinfo* in the project directory during the last download of the application.

After a clean process no online change is possible for the respective application. The program first must be re-downloaded.

7.21.0.4. Clean All

This command, deletes the compile information for all currently active applications. The compile information was created and stored in a file **.compileinfo* in the project directory during the last download of the application.

After a clean process no online change is possible for the respective application. The program first must be re-downloaded.

7.22. Online Menu

Menu to help the user connect to PLC and configure network and protocols settings.

Available commands:

- Login
- Logout
- Create Boot Application
- Download
- Online Change
- Source Download to Connected Device
- Redundancy Configuration
- OPC DA Configuration
- CPU Information
- Reset Warm
- Reset Cold
- Reset Origin
- Simulation
- Security
    - Logoff Current Device User
- Easy Connection
- Clock settings
- Export Online Variables
- Import Online Variables

7.22.1. Login

Default Shortcut: <ALT>+<F8>

This command connects the application to the target device (PLC or simulation target) and thus changes into the online mode.

For a login with the current active application, the code generation must have been completed without errors and the *Communication Settings* of the device must be configured correctly. If the communication settings are not yet set properly, then a dialog box will appear asking you whether you want to set the *active path* by one of the following options (or to cancel the login operation):

- Option 1 (*Yes*): The communication settings dialog should be opened and a network scan performed automatically. If the device recently defined in any project as *active path* (this information is stored on your local system) is found, then this device will automatically be set to be the active one.
- Option 2 (*No*): The communication dialog should be opened without any further automatically executed configuration action. You have to configure the active path manually.

If the *Login* command is called from the *Online* menu, the currently active application will be concerned.

The following situations are possible when going to login with the currently active application (error-free, communication settings configured properly):

- The application is not yet available on the controller: You will be asked you to confirm the download.
- The application project is already available on the controller and has not been modified since the last download, the login will be done without further interaction with the user.
- Another version of the application is already available on the controller in not running mode.
- A version of the application is already available on the controller in RUN mode.
- The application is already available on the controller but has been modified since the last download.

Note: Child application, which have been downloaded to the PLC once and deleted within the device tree during a subsequent logout from the device, will provoke no online change at a repeated login to the device. At least there will be a request if you want to delete them also from the device. E.g. for child applications: Trace.

7.22.1.1. Build Process Before Login

Before Login and if the currently concerned application project has not been compiled since having got opened or since the last modification, it will get compiled. This means the project will be built corresponding to a *build* run in offline mode and additionally compilation code for the PLC will be generated.

If errors are detected during compilation, you will get a message box informing you about that with the following text: *There are compile errors. Do you want to login without download?* You might choose to correct the detected errors first, or to login nevertheless, in this case to that version of the application, which is possibly already available on the controller.

The errors are listed in the *Message* window in category *Build*.

7.22.1.2. Information on the Download Process

When the project gets loaded to the PLC completely at Login or partially at Online Change, then the message window will show information on the generated code size, the size of global data, the needed memory space on the controller and in case of online change also on the concerned POUs.

7.22.2. Logout

Default Shortcut: <CTRL>+<F8>

This command effects a log out of the application. It disconnects the programming system from the target device (PLC or simulation target) and thus changes into the offline mode.

7.22.3. Create Boot Application

This command is available only in offline mode for creating a boot project file. When executed, it opens the *Save boot application* explorer dialog so the user can choose the file's name and which folder it will be saved. The default file's name is <*application name*>.*app*, and the dialog already filter for *Boot applications* *.*app*.

After confirming with Save a further dialog will appear, where you will be asked, whether a possibly already existing compile information file in the project folder should be overwritten.

Choose Yes, if you intend to transfer the new boot application file to the PLC by an external tool, but want to log in to this application later without being forced to perform a new download. This namely would be the case, if due to a previous application download already a compile info file would exist (containing code and reference data of this previous application) which of course would not match with that of the new boot application.

7.22.4. Download

This command is available in online mode. It includes a build and code generation run of the currently active application program. Therefore, besides a syntactical check (build process) also application code will be generated and loaded to the PLC. In the project folder the compile information file <project name>.<device name>.<application ID>.compileinfo will be created.

Note: All variables except for the persistent variables will be re-initialized. Via the object properties dialog you can allocate memory for the application.

If you try to download an application while the same version of this application is already available on the controller, you will get an dialog telling you *Program has not changed. No download will be performed.* The application will not be downloaded to the PLC.

During download, the *Message* window in category *Build* will show a protocol of the running actions (code generation, initialization etc.) and information on the memory areas, code size, global data size and size of allocated memory.

7.22.5. Online Change

If the application project currently running on the controller has been changed in the programming system since it has been downloaded last, just the modified objects of the project will be loaded to the controller while the program keeps running there.

ATTENTION

Online Change modifies the running application program and does not affect a restart process. Make sure that the new application code nevertheless will affect the desired behavior of the system. Depending on the controlled system, damages to machines and parts could result, or even health and life of persons could be endangered.

Notes:

- When an online change is done, the application-specific initializations (homing etc.) will not be executed because the machine keeps its state. For this reason the new program code might not be able to work as desired.
- Pointer variables keep their values form the last cycle. If there is a pointer on a variable, which has changed its size due to an online change, the value will not be correct any longer. Make sure that pointer variables get re-assigned in each cycle.

There are two ways to perform an Online Change:

1. As soon as you try to log in again with a modified application (checked via the *compileinfo*, which has been stored in the project folder during the last download), you will get asked whether you want to do an online change, a download or login without changing.

7.22.6. Source Download to Connected Device

This command downloads the project source code as an project archive to the currently connected controller.

To enable the *Source Download to Connected Device* command, the application must be in online mode. Also, it is necessary to change the project settings to *Download on Demand*. To do this, in the *Project* menu, open *Project Settings*, *Source Download* category, select *Only on demand* and press *OK*.

In *Project Settings*, *Source Download* category, are the default settings regarding the target device, content and time for sending the source code.

7.22.7. Redundancy Configuration

This command allows the user to configure whether the CPU connected will be configured as PLC A, PLC B or Non-Redundant.

You can also configure whether there will be project synchronization between the PLCs, if the redundancy of PLCs is being used.

For further information consult the manual of the corresponding CPU.

7.22.8. OPC DA Configuration

Through this command it's possible to configure the OPC Server installed with the MasterTool IEC XE . For further information about the OPC DA Server configurations check the OPC DA Configuration section.

7.22.9. CPU Information

This command shows a screen with information about the CPU. To access this option it's necessary to select a device as described in the Communication Settings section. If this operation wasn't performed previously, an error message will be displayed. The figure below shows the *CPU Information* dialog screen.

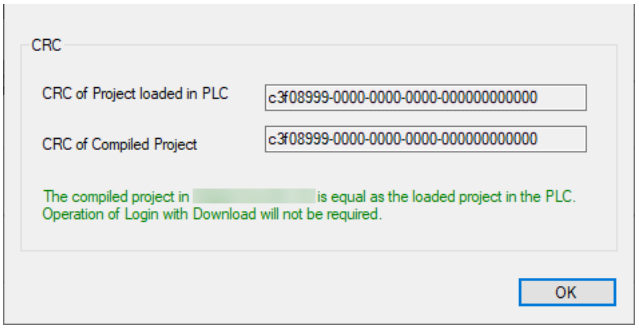

Figure 175: CPU Information dialog

7.22.9.1. CRC

Inside the *CPU Information* dialog, you'll find the CRC of the Project compiled in MasterTool IEC XE and the CRC of the Project loaded in the PLC. By creating a new project without compiling it, the value shown for the CRC of Compiled Project will be always reset. After compiling the project for the first time the CRC value will be shown.

Similarly, if there's no project loaded in the PLC, the value of the CRC of Project Loaded in PLC is going to be reset. By loading a program for the first time the value is going to be shown. If the value from both fields are equal, a message will be shown, informing that the project compiled and the project loaded in the PLC are equal and the Login command can be executed without the need to download. If there are difference between the two fields, it will be informed that it's necessary to perform a download in the PLC since the projects are different.

7.22.10. Reset Warm

This command is available in online mode. It resets – with exception of the remanent (retain or persistent) variables - all variables of the currently active application to their initialization values.

If you have initialized variables with specific values, they will be reset exactly to that value. All other variables are set at a standard initialization value (for example, integers at 0). As a precautionary measure, MasterTool IEC XE asks you to confirm your decision before all of the variables are overwritten. The situation is that which occurs in the event of a power outage or by turning the controller off, then on (warm restart) while the program is running.

A reset disables the breakpoints currently set in the application. If the command *Reset warm* is called during the program run is hold on a breakpoint, the user will be asked whether the cycle should be finished before performing the reset or if the reset shall terminate the task and perform the reset immediately. Be aware, that not all runtime systems are able to perform a reset without finishing the cycle before.

After a reset use the Start command to restart the application.

7.22.11. Reset Cold

This command is available in online mode. It corresponds to *Reset warm*, but besides of normal and persistent variables also sets back retain variables of the currently active application to their initialization values. The situation is that which occurs at the start of a program, which has been downloaded just before, to the PLC (cold start).

A reset disables the breakpoints currently set in the application. If the command *Reset cold* is called during the program run is hold on a breakpoint, the user will be asked whether the cycle should be finished before performing the reset or if the reset shall terminate the task and perform the reset immediately. Be aware, that not all runtime systems are able to perform a reset without finishing the cycle before.

7.22.12. Reset Origin

This command is available in online mode. It resets all variables of the currently active application, including the remanent ones to their initialization values and erases the application on the PLC.

A reset disables the breakpoints currently set in the application. If the command *Reset origin* is called during the program run is hold on a breakpoint, the user will be asked whether the cycle should be finished before performing the reset or if the reset shall terminate the task and perform the reset immediately. Be aware, that not all runtime systems are able to perform a reset without finishing the cycle before.

7.22.13. Simulation

This command is available to switch on and off the simulation mode of the programming system. In simulation mode the application can be run and debugged on a *simulation target* which is always available within the programming system. So no real target device is needed to test the online behavior of an application.

If the command is called from the Online menu, the currently active application will be concerned. If the command is called from the context menu when an application object is selected in the device tree, then the selected application will be affected, no matter whether it is set active.

When command *Simulation* is activated ( ), the device entry in the device tree will be displayed in italic letters and at the first login with the current active application you will be asked whether application *Sim.<device name>.<application name>* should be created and loaded to the simulation target. No communication settings have to be done. See the figure below for an example: Login has just been performed for the currently active application.

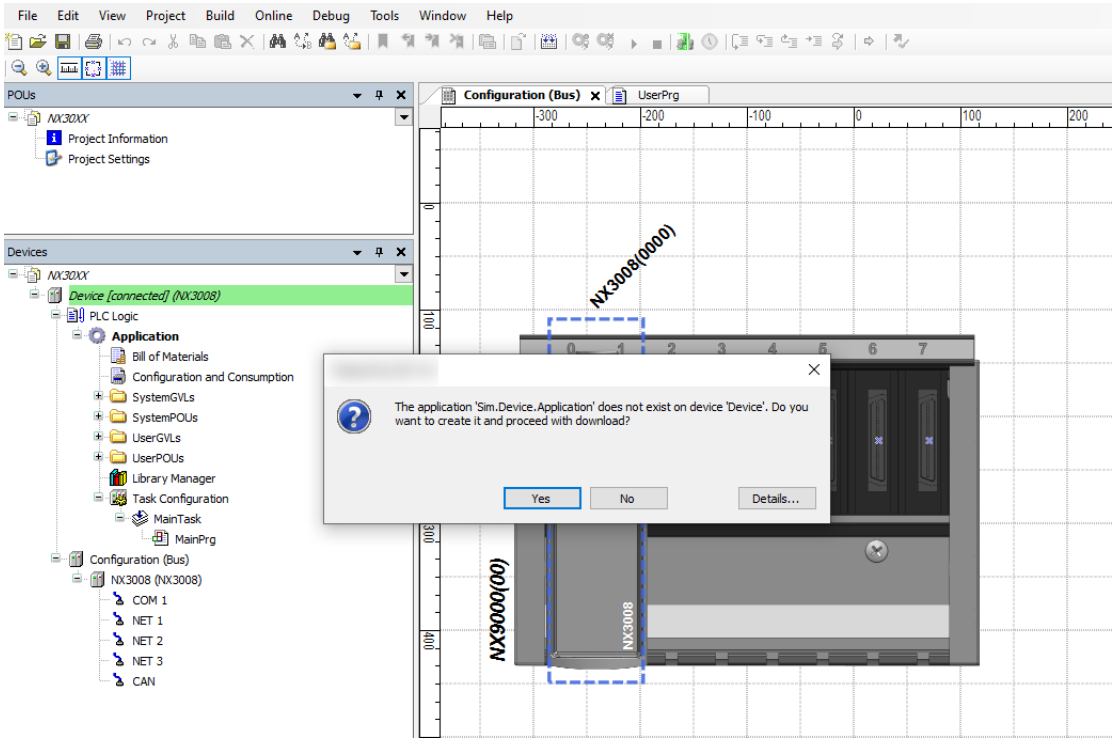

Figure 176: Login to the Simulator Target

After successful login, the user can use the respective online commands to test the application.

To switch off the simulation mode, first log out and then again perform command Simulation. The checkmark in front of the command will disappear, the target entry in the device tree again will be displayed in normal letters and you can log in to a real device.

For further information on simulation, features and restrictions see Simulation Mode

7.22.14. Security

Sub-menu focused on helping the user with security-related tasks.

7.22.14.1. Logoff Current Device User

Symbol: 

When the command is executed, it logs out the user currently logged into the controller. If a connection to the controller through MasterTool IEC XE still exists, it will be disconnected. Note that the application must be in online mode for this action to take place.

7.22.15. Operating Mode

The *Operation Mode* command sets the controller to a state that prevents accidental changes to the project.

You can use these commands, for example, to lock the state of a controller in order to prevent the controller from switching to another state while you program another controller.

When programming is complete, the controller should then be switched to a defined and externally visible state which is set exactly the same way after restarting.

> **WARNING**
>
> This command is to prevent accidental changes. If security requirements are needed, refer to the section User and Access Right Management of the CPU.

When online the  ,  , and  symbols in the status bar indicate the current operating mode. Double-clicking one of these symbols opens a help window.

If the controller supports it, then you can switch the controller to the following operating modes:

-  *Debug*: No restrictions.
-  *Locked*: The current debugging state is locked on the application. It is not possible to set additional breakpoints or force additional variables. Writing variables is still possible, and the breakpoints that are already set remain enabled. Only the *RUN* state of an application is preserved in *Locked* mode, even when the controller is restarted. With this operating mode, developers can be prevented from altering the application on the controller, for example, by setting or deleting a breakpoint, forcing variables, or making changes to the file system. This operating mode is useful to prevent downloading to an incorrect controller when, for instance, multiple controllers of a plant are being programmed.
-  *Operational*: This operating mode ensures that the controller reloads the same applications after a restart and that no debugging features are enabled anymore. The operating mode is set when the controller is fully programmed and must be accepted or is already accepted. Conditions for enabling operational mode: a boot application for each application must exist on the controller, there must be no active breakpoints defined, all applications must be running, there must be no forced values, and additionally, the device may define its own further restrictions.

The *Locked* and *Operational* modes are different in the use cases and in the requirements for enabling the mode. However, for both operating modes the runtime system prevents the following actions:

Regarding the application.

- Download of an application.
- Online Change.
- Force variables.
- Set breakpoints.
- Stop application.
- Reset application.
- Start application.
- Delete application.

Regarding the file transfer of the controller.

- Download of a file to the controller.
- Delete a file on the controller.
- Rename a file on the controller.
- Create a directory on the controller.
- Delete a directory on the controller.
- Rename a directory on the controller.

When the *Locked* or *Operational* mode is active and the user tries to perform a disallowed action, the MasterTool IEC XE will display the following error message (in this case, the CPU was in STOP and the *Operational* mode was selected).

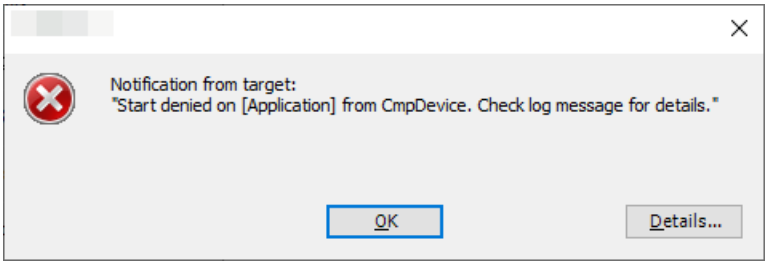

Figure 177: Error message displayed in Operation Mode

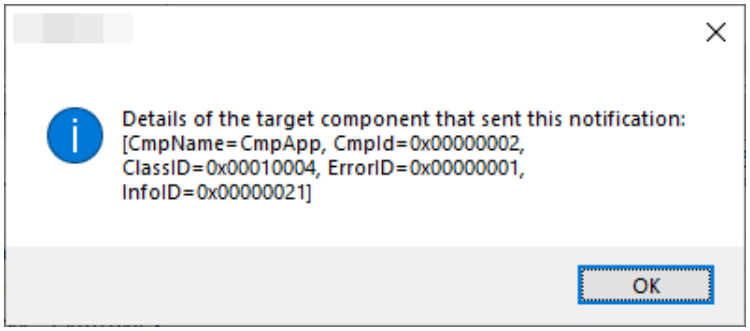

Figure 178: Error details displayed in Operation Mode

If you do not have the original project to log into the CPU and change the operating mode, open any project or create a new one, then select the CPU in the *Gateway* and access the *Communication / Operating Mode* menu. Next, select the only available option, '*Debug*' to change the *Operating Mode* state. To change the operating mode from *Debug* to *Locked* or to *Operational*, you must be logged into the CPU.

7.22.16. Easy Connection

Symbol: 

The *Easy Connection* command opens the *Easy Connection* dialog. This screen searches for and displays all PLCs connected to the network, so that the user can make changes to their network settings.

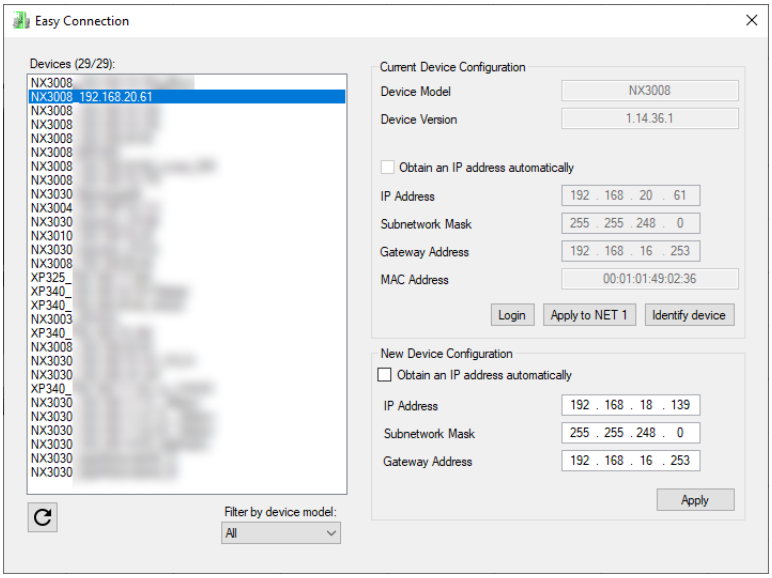

Figure 179: Easy Connection Dialog

In the group on the left, called *Devices,* it's possible to see all the PLCs available on the network. When one is selected, the elements on the right will be unlocked.

Under *Current Device Configuration* there's the device's current information: *Device Model*, *Device Version* (Firmware Version), *IP Address*, *Subnetwork Address*, *Gateway Address* and *MAC Address*. There is also the *Identify Device* button, which, for a few seconds, activates a function to find your PLC visually.

The *Login* and *Apply to NET 1* buttons can only be used when a project is open. There's the buttons descriptions:

- *Login*: Logs directly into the selected PLC
- *Apply to NET 1*: Applies the settings of the selected PLC to NET 1

Unlike *Current Device Configuration*, in the *New Device Configuration* group you can define new settings for the device. After defining new settings for *IP Address*, *Subnetwork Address* or *Gateway Address* and clicking *Apply*, these settings will be applied to the device. You can click on the *Obtain an IP Address automatically* checkbox to obtain a network configuration automatically, now without having to have a project open.

- *Obtain an IP address automatically*: Gets a valid IP address automatically. If this is selected, isn't necessary to configure an *IP Address*, *Subnetwork Address* or *Gateway Address*.

You can filter devices by model using the *Filter by device model* filter. Also, it's possible to reload the list of devices by clicking on the ( ) button.

Note: The default name for the PLC is <*PLC model*>_<*IP address*>.

7.22.17. Clock settings

Symbol:

The command opens the dialog *Clock Settings*. For the command to be enabled, the PLC need to be selected in the gateway.

The dialog shows the current date-time in the PLC. The user can change this time by the current date-time of the computer or select a date, starting from the year 2000.

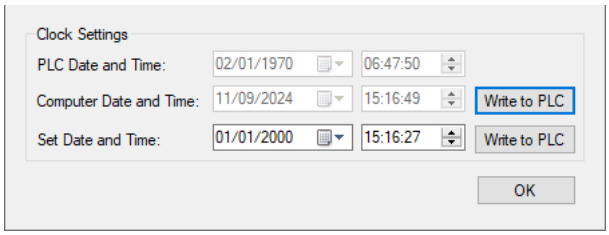

Figure 180: Clock Settings Dialog

7.22.18. Export Online Variables

The *Export Online Variables* command allows you to export the values of the variables declared in POUs and GVLs to a file named <Project name>_ExportedVariables. The file is a compiled file. This feature makes it possible to save the configuration state of the application at a given time, since it allows to restore it to that state using the *Import Online Variables* command.

This is very useful when the settings and configuration values are not stored in an HMI or SCADA system that runs the application. In case of application failure, CPU failure or any other hardware element that causes loss of information, the created file can be used to restore the values after reloading the project into the PLC.

The file is created in the directory where the project is opened. Running the *Export Online Variables* command displays the screen shown in the figure below.

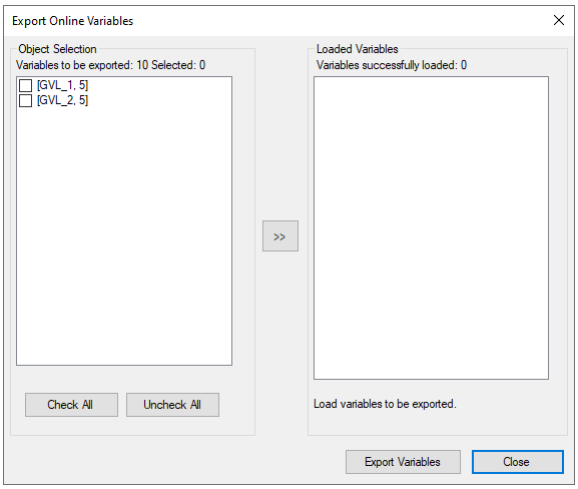

Figure 181: Export Online Variables Screen

On the left side of the screen all the GVLs available in the project are displayed, allowing you to select which ones will have their values read and saved to a file. It's possible to configure individually all the objects that will have their values exported or, to simplify the editing process, it's also possible to use the *Check All* and *Uncheck All* buttons.

After selecting the objects to be exported and pressing the » button in the center of the screen, all the variable contents and their respective values will be stored by MasterTool IEC XE in internal structures of the tool. The screen shown in the figure below will be displayed and all the variables present in the selected objects will be on the right side of the screen.

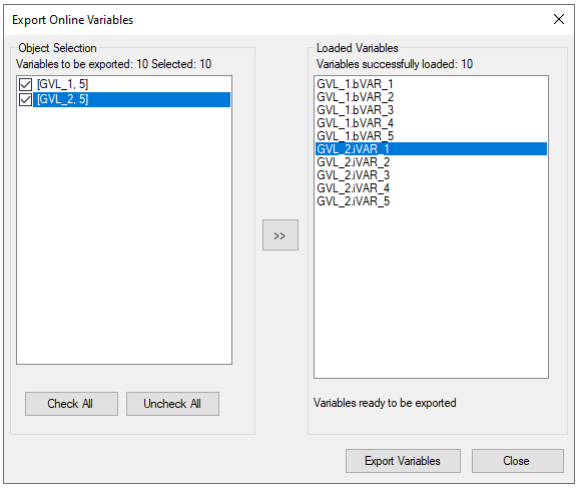

Figure 182: Variables to be Exported by the Export Online Variables Command

Pressing the *Export Variables* button will save the contents to a file created in the project directory. If the file already exists, it will be modified to contain only the contents of the last export action performed.

This command is only available after a command to *Login* to a CPU has been executed, i.e. it's necessary to be online for the command to appear in the menu. The same monitoring mechanism that is used to monitor variables in GVLs, POUs and monitoring windows is also used to load the data to be stored by the command. This mechanism has a limit on the amount of data that can be read from an open instance of MasterTool IEC XE . The limit is 60,000 elements.

Therefore, it's not possible to store more than 60,000 elements in a file created with this command. If you select a set of GVLs in which the sum of the elements is greater than 60,000, MasterTool IEC XE will inform you that the limit has been exceeded and the operation won't be performed.

In the case of simple variables, each variable represents one element. In the case of types such as Structs and Arrays, each element represents a read value that is counted in this limit. Beyond this limit on the maximum number of elements, the array data type is limited by a maximum number of bytes. Arrays larger than 60,000 bytes can't be exported. This limit is checked by the tool.

Exporting and importing large amounts of data (close to the 60,000 limit) consumes shared resources with MasterTool IEC XE monitoring tool. These resources may be exhausted during program execution. In this case, these resources can only be restored by restarting the program.

ATTENTION

Considering the limit of variables that can be exported, it is important to define a strategy for declaring them. This will make it easier to select the variables that are really important and need to be saved. Variables that define customizations or application configurations must be saved to allow their restoration. Variables that are cyclically recalculated by the PLC logic don't need to have their values saved.

ATTENTION

This feature allows operations in simulation mode. However, the limits of this operation in this mode are smaller than in a real PLC - i.e. it is not possible to perform operations with 60,000 elements, and MasterTool IEC XE won't do it.

A good way to store the values of a particular project is to create a *Project Archive*. This allows saved variables to be saved along with application backups. To create a *Project Archive*, go to *File* > *Project Archive* > *Save/Send Archive*. The following figure shows the Project Archive screen.

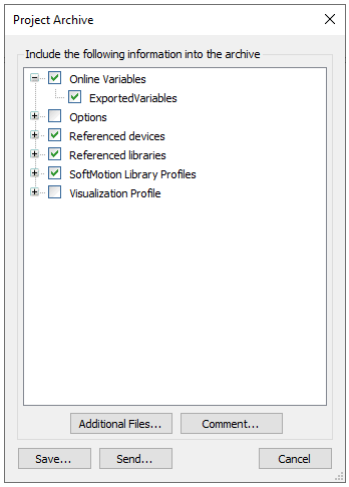

Figure 183: Online Variables in Project Archive

If the *Online Variables* option is checked, once the project archive is created, it can be extracted and the stored values can be extracted along with the application.

7.22.19. Import Online Variables

The *Import Online Variables* command lets you import values exported with the *Export Online Variables* command. The file to be imported must be in the same folder as the project file and named: <Project name>_ExportedVariables. This command is only available after logging into the CPU. To use this command, you need to be online.

To enable this, the file must be in the project folder or extracted from a project archive. To do so, go to *File* > *Project Archive* > *Extract Archive*. Make sure the *Online Variables* option is selected.

If you edit the project before importing, there may be inconsistencies between the exported file and the project. See the list below for details on how to handle these situations.

- Variable present in the exported file, but removed from the project: the removed variable won't be written and MasterTool IEC XE will inform which variables couldn't be imported by the end of the process.
- Variable added to the project, but absent from the exported file: the added variable will keep its value after importing.
- Exported variable's type is different from the project variable: the variable whose type has been altered won't be written and MasterTool IEC XE will inform which variables couldn't be imported by the end of the process.
- String type variable has its size modified: if the string in the project is equal or larger than the one exported, it will be imported; otherwise, it won't be written and MasterTool IEC XE will inform which variables couldn't be imported by the end of the process.
- Struct Element or Array Position present in the exported file, but removed from the project: the removed variable won't be written and MasterTool IEC XE will inform which variables couldn't be imported by the end of the process.
- Struct Element or Array Position added to the project but absent from the exported file: the added variable will keep its value after importing.

If there are no restrictions to the import, MasterTool IEC XE will display a message when the process is done successfully. If there are any restrictions, a window with all variables that were not imported can be consulted by the end of the process.

7.23. Debug Menu

Provides commands to start and stop the program in PLC, in addition to running breakpoint features for testing purposes and force values. Most of these commands can also be used in simulation mode.

Available commands:

- Start
- Stop
- New Breakpoint..
- New Data Breakpoint..
- Edit Breakpoint..
- Toggle Breakpoint
- Disable Breakpoint
- Enable Breakpoint
- Step Over
- Step Into
- Step Out
- Run to Cursor
- Set Next Statement
- Show Next Statement
- Write Values
- Force Values
- Unforce Values
- Display Mode
- Create PLC Crash Report

7.23.1. Start

Symbol: 

Default Shortcut: <F5>

This command starts the application program on the device (PLC).

7.23.2. Stop

Symbol: 

Default Shortcut: <SHIFT>+<F8>

This command stops the application program on the device (PLC).

7.23.3. Breakpoints

A breakpoint, set in an application program, will cause a break during the execution of the program. The possible breakpoint positions depend on the editor. In each case, there is a breakpoint at the end of a POU. See Breakpoints for a description of the commands concerning breakpoints.

7.23.4. New Breakpoint...

Symbol: 

This command is used to insert a new breakpoint in one of the POUs. It does not care where the cursor currently is placed, the *New Breakpoint* dialog will open where in sub-dialog *Location* you can choose one of the possible breakpoint positions all over the project and in sub-dialog *Condition* can define some conditions for the new breakpoint.

Note: For setting a breakpoint at the current cursor position notice command Toggle Breakpoint

7.23.4.1. Location

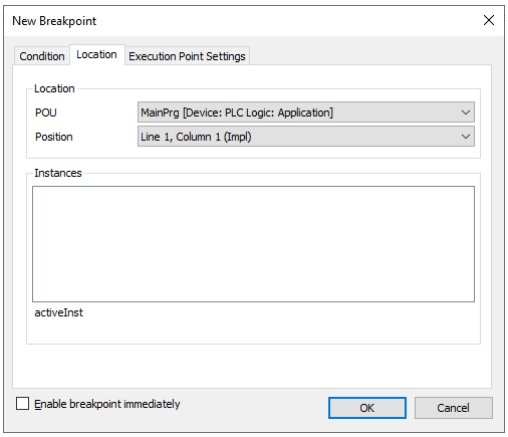

Figure 184: Location Dialog

- *POU*: The selection list offers all POUs currently available in the project. Select one to set a breakpoint.
- *Position*: The selection list offers all possible breakpoint positions of the currently selected POU. Depending on the editor type these positions are defined by Line+Column numbers (text editors) or as Network or Element numbers (graphic editors). In case of a function block additionally the user has to decide whether the breakpoint should be set in the implementation or in an instance. If it should be set in the implementation, leave option 'Instance Path' deactivated. If it should be set in an instance, activate option 'Instance Path' to select the desired instance; see the following.
- *Enable Breakpoint Immediately*: Breakpoints are created disabled by default. This checkbox enable then already when created.

7.23.4.2. Condition

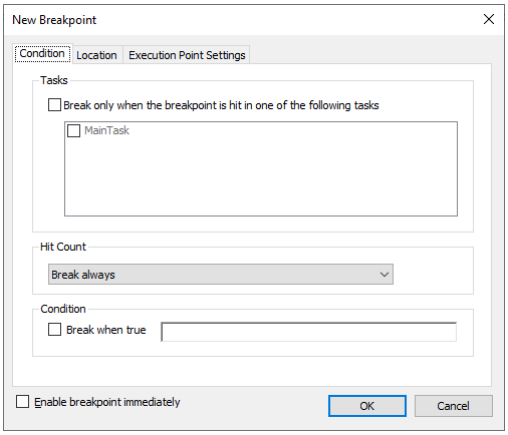

Figure 185: New Breakpoint Dialog Condition

- *Break only when the breakpoint is hit in one of the following tasks*: Activate this option if the breakpoint only should be effective, if the POU where it is placed, is processed by particular tasks. All tasks currently defined in the project will be listed and the desired one(s) can be defined by setting tick(s) correspondingly.

Options of *Hit Count* group:

- *Break always*: The program always will stop at the breakpoint.

In addition to that, there is the possibility that the program do not stop at the breakpoint until the breakpoint has been hit the number of times defined (Enter the desired number resp. select it in the number field):

- *Break when the hit count is equal to*
- *Break when the hit count is a multiple of*
- *Break when the hit count is greater than or equal to*

7.23.4.3. Execution Point Settings

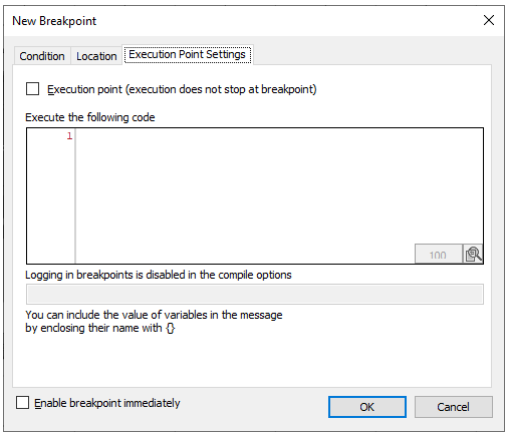

Figure 186: Execution Point Settings Dialog

- *Execution point (execution does not stop at dialog)*: When checked, allows to write a code that will be executed every time the breakpoint is hit.

7.23.4.4. Breakpoint Positions

The possible breakpoint positions depend on the editor. Basically they are those positions in a POU at which values of variables can change or at which the program flow branches out resp. another POU is called.

Note: Breakpoints in methods: A breakpoint will be set automatically in all methods, which might be called. Therefore, if an interface-managed method is called, breakpoints will be set in all methods of function blocks implementing that interface and also in all derivative function blocks subscribing the method. If a method is called via a pointer on a function block, breakpoints will be set in the method of the function block and in all derivative function blocks, which are subscribing the method.

7.23.4.5. Breakpoint Symbols

The figures below shows the possible symbols assumed by breakpoints.

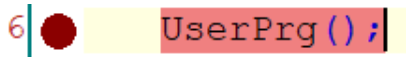

Figure 187: Breakpoint in Online Mode

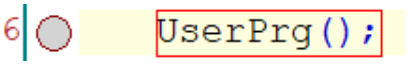

Figure 188: Disabled Breakpoint

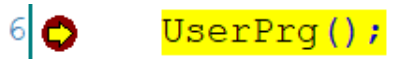

Figure 189: Program Stop at Breakpoint

7.23.5. New Data Breakpoint...

Symbol: 

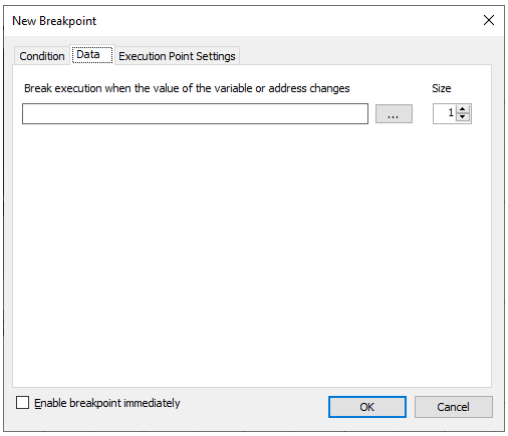

Figure 190: New Breakpoint Dialog Data

Open the *New Breakpoint*, as the New Breakpoint.. does. The difference is that *New Data Breakpoint* has the *Data* sub-dialog, where the user can select a variable and how much the variable value (or address) should change till the execution breaks.

7.23.6. Edit Breakpoint...

Symbol: 

The command opens the *Breakpoint Properties* dialog, which contains the same three subdialogs as New Breakpoint..: *Condition*, *Location*, and *Execution Point Settings*, which allow the user to change the settings of the selected breakpoint.

Note: For a breakpoint to be selected, the cursor must be in the same as it.

7.23.7. Toggle Breakpoint

Default Shortcut: <F9>

This command basically toggles between status *enabled* and *disabled* of a breakpoint. However, it also effects that a new breakpoint will be set if not yet one is already set at the current breakpoint position.

Note: Each active breakpoint will get an *inactive* one, when you leave online mode and login again afterwards.

7.23.8. Disable Breakpoint

Symbol: 

The command disables an enabled breakpoint. To select an breakpoint, it's necessary the cursor halted at it.

7.23.9. Enable Breakpoint

Symbol: 

The command enables an disable breakpoint. To select an disabled breakpoint, it's necessary the cursor halted at it.

7.23.10. Step Over

Symbol: 

Default Shortcut: <F10>

This command can be used for stepping through a program in online mode, for example for debugging purposes. For instructions on just one level, this command is equivalent to a step by step. If however a POU call is reached, then *Step Over* effects that this POU will be executed completely within the current step. In SFC, a complete action will be executed.

If you would like to jump to the first instruction of a called POU, you had to use the *Step Into* command.

7.23.11. Step Into

Symbol: 

Default Shortcut: <F8>

This command can be used for stepping through a program in online mode, for example for debugging purposes.

A single step will be executed. The program will stop before the next instruction. If necessary, there will be a changeover to an open POU. If the present position is a call-up of a function or of a function block, then the command will proceed on to the first instruction in the called POU.

In the possible halt positions during stepping depend on the editor. The current position is indicated by a yellow shading.

All other situations, the command will act like *Step Over*.

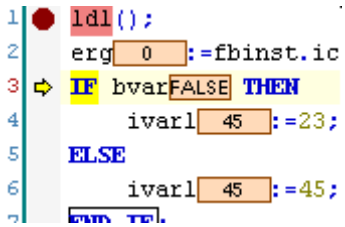

Figure 191: Step Into

7.23.12. Step Out

Symbol: 

Default Shortcut: <SHIFT>+<F10>

This command can be used when stepping through a program in online mode, for example for debugging purposes.

If the application program does not contain any calls, *Step Out* will effect a jump back to the start of the application. If however you previously have stepped into a called POU, *Step Out* will effect a jump back just to the calling instruction. Therefore, in case of nested calls, *Step Out* will lead you back through the hierarchy of callers step by step. This for examples allows to go back and to step in to another called POU.

7.23.13. Run to Cursor

Symbol: 

This command can be used in order to execute the program up to a temporarily definable position.

This corresponds to setting a breakpoint at the desired next stop position and stepping there.

Place the cursor at the desired *next break* position and perform the command. The instructions between the current break position and the cursor position will be executed.

7.23.14. Set Next Statement

Symbol: 

This command can be used when stepping through a program in online mode.

It defines the next instruction (statement) to be executed. For this purpose place the cursor in the desired statement and perform the command.

7.23.15. Show Next Statement

Symbol: 

This command can be used in online mode to get back to the current execution position.

This might be useful, if for any reason have left the window where you are currently debugging, and have put the cursor somewhere else in the programming system. The window showing the respective POU will be put again in foreground and the cursor will be placed again at the current position of execution.

7.23.16. Write Values

Default Shortcut: <CTRL>+<F7>

ATTENTION

Extraordinary changes of variable values in an application currently running on a controller can cause unwanted behavior of the controlled system. Evaluate possible dangers before Writing or Forcing of variables and make appropriate security precautions. Depending on the controlled system, damages to machines and parts could result, or even health and life of persons could be endangered.

To write a value by this command means to set a variable on the PLC to a defined value at the beginning of the next execution cycle. The command effects all variables of the currently active application, which are prepared for writing.

To prepare variables for writing, the desired value must be defined in online mode in one of the following places, which are used for monitoring:

- In a watch view defined in the project, containing a list of variables to be monitored (watch list);
- In the online view of the object within the declaration part of the respective editor.

Note: See in this context also the Force Values command for permanently setting a defined value.

Example:

Open an object in online mode, for example a program written in ST. In the declaration part of the editor window you will find the displayable expressions listed in a table. Click on the concerned field in column *Prepared Value*, enter the desired value. Perform command *Write Values* that is by default part of the Online menu. The value will immediately be written to the respective field in column *Value*, which means that it is written to the controller. The *Prepared Value* field will be empty again.

The same can be processed in a watch view containing the desired expression.

Notice in this context the *Prepare Value* dialog, which is available for currently forced variables and where a new value to be written can be defined.

7.23.17. Force Values

Default Shortcut: <F7>

ATTENTION

Extraordinary changes of variable values in an application currently running on a controller can cause unwanted behavior of the controlled system. Evaluate possible dangers before Writing or Forcing of variables and make appropriate security precautions. Depending on the controlled system, damages to machines and parts could result, or even health and life of persons could be endangered.

This command is available in online mode. It effects that one or more variables of the currently active application are permanently set to user-defined values in the PLC. The setting will be done both at the beginning and at the end of a cycle. The figure below presents the sequence of processing in a cycle of application program.

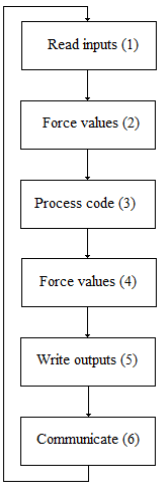

Figure 192: Sequence of Processing in a cycle

If a forced variable has its value changed during the code execution (3), its value will keep changed until the follow force values process. This kind of situation can generate some inconsistencies during the logic monitoring. It happens because the monitored value can be different of the real one used by the application program, since the communication is treated only in the end of CPU sequence of processing.

Note: See in this context also the Write Values command for setting a defined value only once at the beginning of a cycle.

The forcing will remain active until it is explicitly suspended by the user for particular and for all variables, or until the application gets logged-out.

ATTENTION

The forcing operation does not actuate in %I or %Q operands updated with the functions REFRESH_INPUT and REFRESH_OUTPUT. These functions execute a read to %I operands or a written from %Q operands after executing them and they do not consider the forcing effects. For this reason, it is not recommended forcing operand that have been updated by the functions REFRESH_INPUT and REFRESH_OUTPUT that are active in the application program.

To prepare variables for forcing, the desired value must be defined in online mode in one of the following places, which are used for monitoring:

- In a watch view defined in the project, containing a list of variables to be monitored (watch list).
- In the online view of the object within the declaration part of the respective editor.
- In the online view of the object within the implementation part of the FBD or LD editor.

A forced value is indicated with a symbol F.

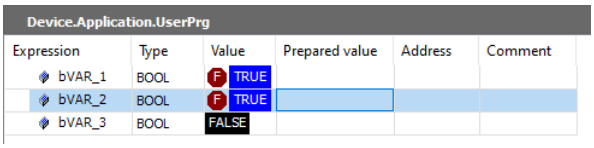

Figure 193: Forced Values in the Declaration Editor of a POU (Online View)

7.23.17.1. Prepare Value Dialog

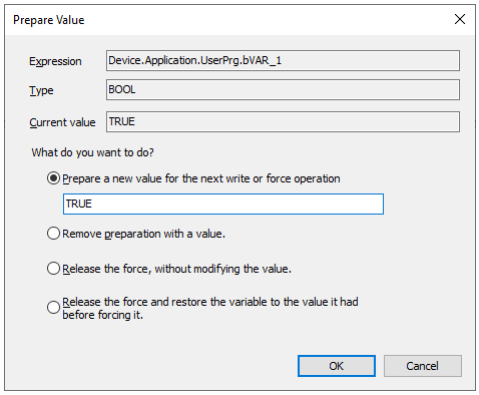

Figure 194: Prepare Value Dialog

This dialog is used to prepare a new value for a variable, to remove a prepared value, to release a forced variable or to release it and additionally reset its value to the one the variable was assigned to before forcing.

The dialog will open if you click in the *Prepared Value* field of a currently forced value or in the inline monitoring field of the variable in the implementation part of the FBD/LD editor.

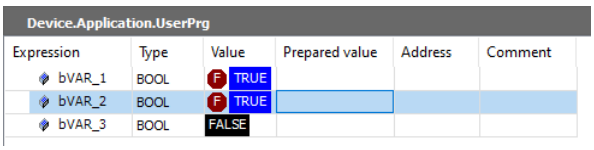

Figure 195: Forced Values in the Declaration Editor of a POU (Online View)

The following information on the currently concerned variable is displayed:

- Expression: path of the variable (*Device.Application.UserPrg.bVar_1*, for example).
- Type: data type (*BOOL* for example).
- Value: *TRUE* or *1*, for example.

Choose one of the following options concerning *What do you want to do?*:

- *Prepare a new value for the next write or force operation*: Depending on the data type of the variable you can enter a new number or string which you want to get assigned to the variable.
- *Remove preparation with a value*: The prepared value for a variable will be removed.
- *Release the force, without modifying the value*: The variable will be marked as <Unforce> and thus is prepared to get the current value read from the PLC.
- *Release the force and restore the variable to the value it had before forcing it*: The variable will be marked as <Unforce and restore> and thus is prepared to get the value it had before forcing.

According to the chosen option, after leaving the dialog with *OK*, in the *Prepared Value* field of the monitoring view the variable will show a new value or <*Unforce*> or <*Unforce and restore*>. At the next *Force Values* and *Write Values* (for the first option) command the prepared values will be set.

MasterTool IEC XE has a maximum limit on the number of allowed forces. This limit is 128 forces, i.e. 128 forcing entries regardless of the type of the variable. For example, if it is forced on a variable of type BOOL, this action will consume one forcing entry, as well, will be consumed only one entry if forced a REAL. After forcing 128 entries, the MasterTool IEC XE does not allow forcing, showing a warning. In addition, it is only possible to force symbolic variables. For the forcing of direct representation variables, these must be associated with a symbolic variable in its Declaration, or be declared in a local bus editor as inputs and outputs of a module and MODBUS mappings.

It is also not allowed forcing device diagnostics, even if they are mapped into symbolic variables. This is not allowed to prevent a wrong interpretation of diagnostic devices. In case you need to write a value in an area of diagnostics it must be used the command *Write Values* (not *Force Values*).

In addition to the limit of the number of entries in the list of forces, is also considered, before forcing a new entry, if this will not exceed the limit of forcing buffer. This limit is 10240 bytes. For basic types variables occurs not overflow buffer limit, however, in the case of variables of complex types, such as STRINGs, when they are forced, will be considered if the amount of data bytes forced is not overflowing the buffer limit.

7.23.18. Unforce Values

Default Shortcut: <ALT>+<F7>

This command is available in online mode. It serves to release the forcing for all variables of the currently active application.

The variables will get the current value read from the PLC. This corresponds to option *Release the force, without modifying the value*, which can be activated in the *Prepare Value* dialog for a forced variable.

7.23.19. Add All Forces to the Watch List

This command is available in online mode when one of the watch views of watch 1, 2, 3 or 4 is active, it can be viewed in context menu when click in the *Watch View*. It serves to add all currently prepared or already forced variables of the active application to this watch list. Notice however that this works only for docked watch views.

Notice also the possibility to use the watch view *Watch all Forces*, which automatically contains all currently prepared or forced values and additionally provides commands for releasing the forces.

7.23.20. Display Mode

These commands can be used to choose which of the following formats should be used for the monitoring display. Perform a mouse-click on the desired option in the Display Mode submenu. The currently set option is marked by a tick.

- Binary
- Decimal
- Hexadicaml

7.23.21. Create PLC Crash Report

The command can only be executed when the program is in online mode. Other requirements are: the program needs to be hold in a breakpoint, or the PLC has to be threw a exception.

When the command is executed with this condition, an *Project Archive* is generated with all necessary information for a debug.

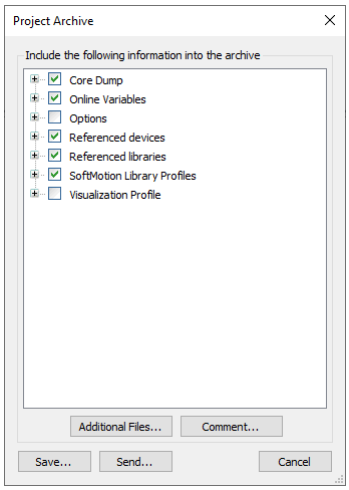

Figure 196: *Project Archive* from *Create PLC Crash Report*

7.24. Tools Menu

Menu to help the user to install external elements (devices, libraries, etc), manipulate the license settings and adjust some PLC and object settings.

The available commands are:

- Library Repository
- Device Repository
- License Repository
- OPC UA Information Model Repository
- License Manager
- Start PACTware
- Options..
  - CFC Editor
  - Debugging
  - Declaration Editor
  - Device Description Download
  - Device Editor
  - FBD, LD and IL Editor
  - International Settings
  - Load and Save
  - PLCopenXML
  - Proxy Settings
  - Refactoring
  - SFC editor
  - SmartCoding
  - Text Editor

- Visualization
- Visualization Styles
- Visualization User Management

- Miscellaneous
- Scripting

7.24.1. Library Repository

Symbol: 

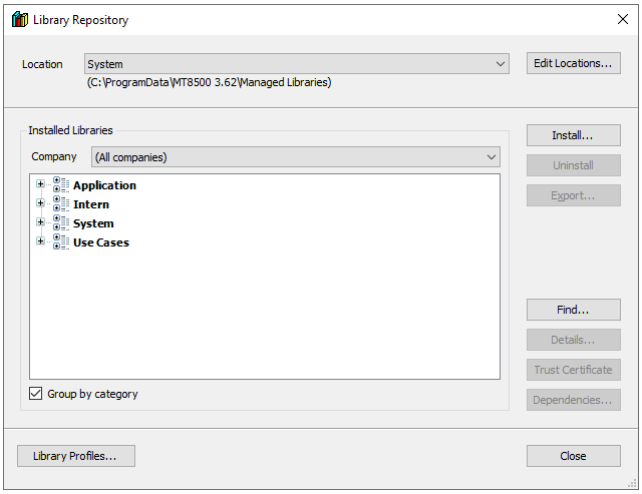

Figure 197: Library Repository Dialog

The dialog shows the currently installed libraries, as well as their locations (repositories). Repositories can be added, modified or deleted in this dialogue and libraries can be installed or uninstalled.

As the company and the location (directory on the local system where the library files are stored) currently selected, all the libraries installed will be displayed in a list. This list shows the names (title), the version number and the name of the company, as provided by the project information library.

The option *Group by category* shows the listing sorted by categories of libraries, where the names are displayed in the form of folders that can be opened or closed to show or hide the libraries respectively.

If this option is not enabled, the libraries are listed alphabetically.

According to the available buttons, see the descriptions of:

- Library Repository (Edit locations)
- Installation and uninstallation of libraries (Install and Uninstall)
- Further information about libraries (Details and Dependencies)

7.24.1.1. Edit Locations

Multiple repositories can be used to manage libraries. All repositories currently defined are shown in the selection list in *Location*. By default, the location described as *System* is always available as defined in MasterTool IEC XE installation.

To edit the path or name of the repositories, use the *Edit Locations* button and open the corresponding dialog.

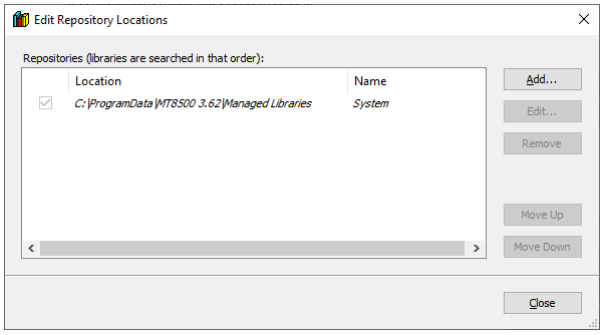

Figure 198: Edit Repository Locations Dialog

Currently defined locations are listed in the *Repositories* window. The search order is from top to bottom. To modify this order use the buttons *Move up* or *Move down*.

<*All locations*> displays all currently defined libraries locations. In this view it is not possible to perform an installation.

7.24.1.1.1. Define new repository and Change name and/or Path from a repository

To add a new repository, use the *Add* button. The *Repository Location* dialog will be opened and in the location field should enter the path of the new repository. To do this, use the button ... and look for a new folder properly. Note that the folder must be empty. In the Name field, type a symbolic name for the location.

To modify an existing repository, select the respective item in the dialog and use the *Edit* button. The *Repository Location* dialog will also be open to edit the path and the name of the same.

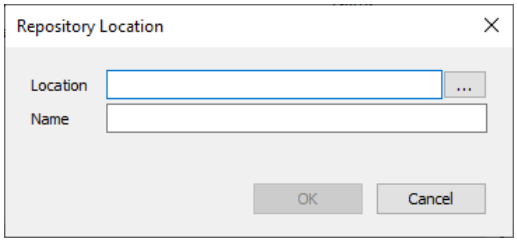

Figure 199: Add or Edit a Repository

Note: Only empty folders can be characterized as a repository. The repository of the *System* is not editable.

7.24.1.1.2. Deleting an existing repository

When you select an item in the repository list and use the delete button, the user must point at which item he wants to delete. He can also delete the entire folder containing system library files.

7.24.1.2. Installation and Uninstallation of Libraries

Only a library installed on the local system (Library Repository) can be included in a project. The information in the project should include the title, version and, optionally, the name of the library company.

To install a library, select the repository in which it must be added, and then press the *Install...* button.

The *Select Library* dialog will open the standard dialog to search for files. Select the library you want, and then close the dialog. The library will be added to the list of libraries currently installed on the *Library Repository* dialog.

If the user chooses a library that cannot be installed due to mandatory information (title and version) he will get an error message.

To uninstall a library, select it in the list of installed libraries in corresponding dialog and use the *Uninstall* button.

7.24.1.3. Further information about specific libraries

The *Details* button for the currently selected library provides detailed information - title, version, company, file size, date of creation, date of modification, last access date, attributes.

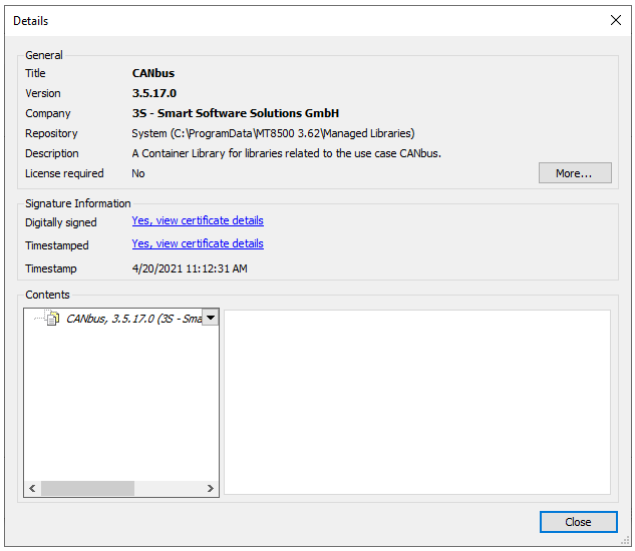

Figure 200: Details Dialog

The *Dependencies...* button displays the current library dependencies (other libraries included in this) with the following information, title, version and company.

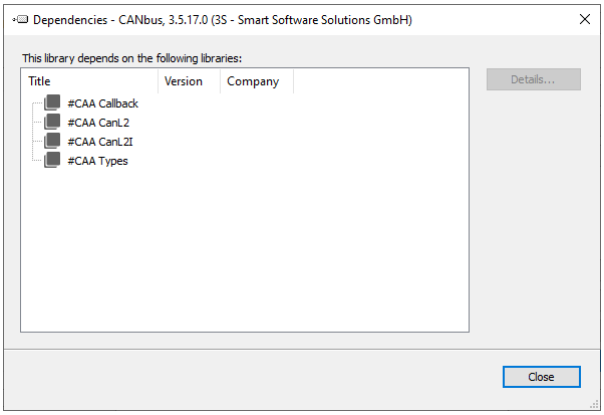

Figure 201: Dependencies Dialog

7.24.2. Device Repository

Symbol: 

The command opens the *Device Repository* dialog, which is used to manage devices installed on the local system and integrate them into MasterTool IEC XE projects.

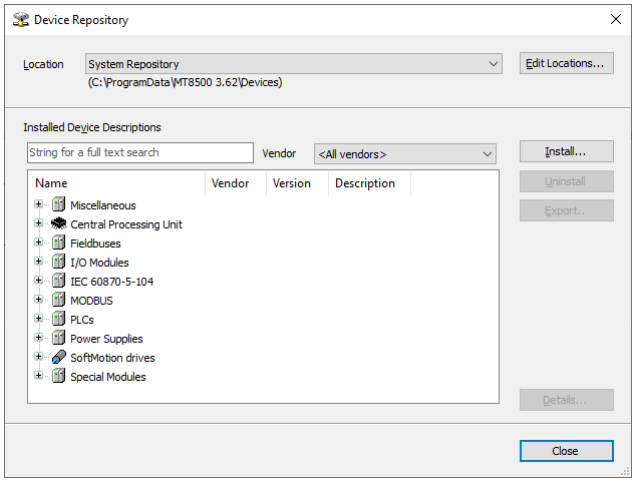

Figure 202: Device Repository

- *Location*: The device repository directories on the local system are displayed, with the list box showing the currently configured locations. By default, MasterTool IEC XE creates the system repository during installation. The devices from the selected location are listed in the *Installed Device Descriptions* area.
- *Edit Locations...*: Opens the *Edit Repository Locations* dialog.

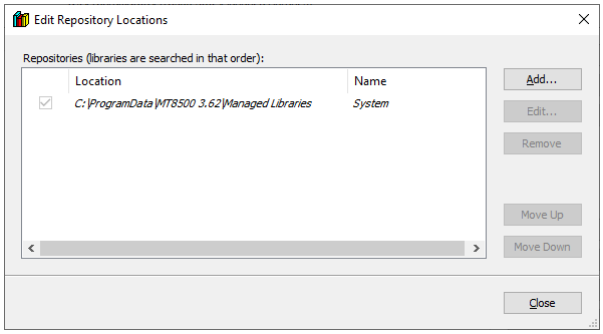

Figure 203: Edit Repository Locations Dialog

- *Add*: A new repository is created by opening the *Repository Location* dialog. The selected directory in the *Location* input field must either be empty or a valid repository.
- *Edit*: Opens the *Repository Location* dialog (refer to *Add*).
- *Remove*: A dialog prompt appears, allowing you to choose whether to delete the respective directory from the hard disk.

7.24.3. License Repository

Symbol: 

The command opens the *License Repository* dialog, allowing you to view information about individual licenses. For this feature to function, MasterTool IEC XE must be in either offline or online mode. Within the license repository, you can access information from the central license server related to licenses based on ticket numbers. These ticket numbers can be provided directly from the clipboard or by importing a text file.

- *Tickets*: A list of the ticket IDs imported into the repository for licensed components.
- *Licenses*: List of selected tickets. Also display the name and status of the tickets. The status can be:
  -  : The license is available and valid.
  -  : The license was found, but it is invalid.
  -  : The license was not found.

7.24.4. OPC UA Information Model Repository

Symbol:

The command opens the *OPC UA Information Models* dialog, where you can manage OPC UA models installed on the local system. These models can be integrated into MasterTool IEC XE projects directly through the dialog.

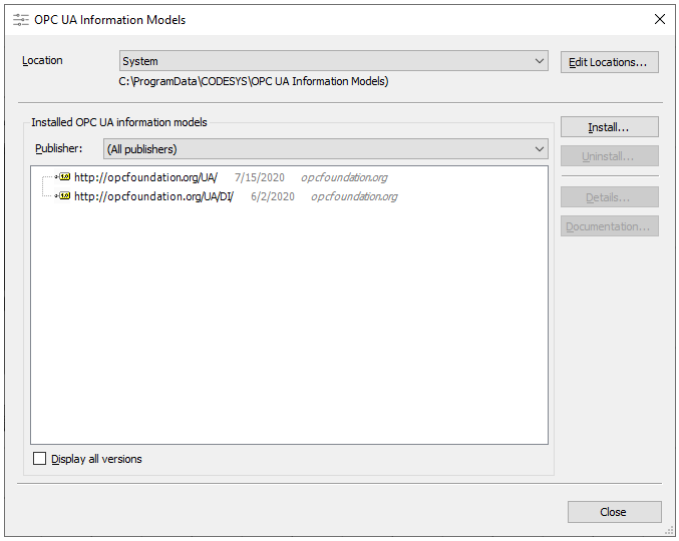

Figure 204: OPC UA Information Models Dialog

- *Location*: The OPC UA information model directories on the local system are displayed, showing the currently set locations. By default, MasterTool IEC XE creates the system repository during installation, and the information models from the selected location are listed in the *Installed OPC UA Information Models* section.
- *Edit Locations*: Opens the *Edit Repository Locations* dialog.
- *Installed OPC UA information models*: The list of installed information models is displayed, and double-clicking opens the documentation for each model. Note that the information models from this repository can also be included in project archives.
- *Install*: The command opens the *Select Installed OPC UA Information Model(s)* dialog. The file type is set to OPC UA Information Models (*.NodeSet2.xml), such as *Informationmodel.NodeSet2.xml.* When you click *Open*, the selected information model is added to the repository. Alternatively, you can select files of any type (*.*), such as OPC UA documentation in formats like PDF or Word. Upon clicking *Open*, the *Assign Documentation OPC UA Information Models* dialog appears.
- *Uninstall*: Uninstalls the selected OPC UA information model
- *Details*: Opens the *Details* dialog for the selected information model, containing additional information about the OPC UA Information Model, like *Model URI*, *Publication date*, *Publisher*, *Repository*, *Alias*, *Documentation available* and *Install documentation*, where it's opened *Select OPC UA Information Models Documentation* and the OCP UA Information Model Documentation (*.pdf) data type is set to default in the dialog.
- *Documentation*: The command opens the installed documentation for the selected information model. If no documentation is available for the selected model, the command is disabled.
- *Display all versions*: All installed versions of the information model are displayed in a hierarchical tree structure.

7.24.5. License Manager

Symbol:

The command opens the wizard for configuring licenses for CODESYS add-on products, beginning with the *License Manager – Select Target* dialog. The *License Manager* manages licenses for CODESYS add-on products on the local computer and runtime add-on products on devices, supporting both soft container installations and dongle-based licenses.

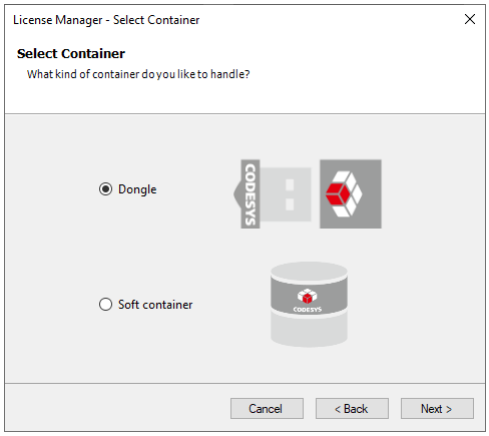

Figure 205: License Manager Select Target Dialog

In the *License Manager – Select Target*, where you decide where the license will be installed, the user can choose between *Workstation* or *Device*. In *Workstation* it will be installed in the local computer, and *Device* the license will be installed in the controller, so the connection to this device need to be configured configured correctly in order to the licensing operation.

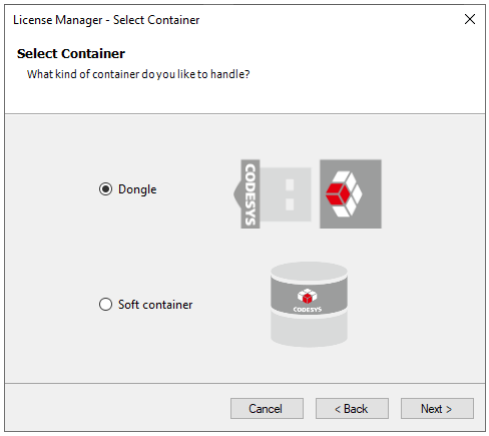

Figure 206: License Manager Select Target Dialog

The *License Manager – Select Container*, where you select the container type of the license, the user can choose between *Dongle* and *Soft container*. In *Dongle*, a corresponding dongle has to be connected to the computer or device. In *Soft container*, a corresponding soft container has to be registered in the *CodeMeter Control Center*. The MasterTool IEC XE installation provides a soft container.

Note: Not all devices support dongles.

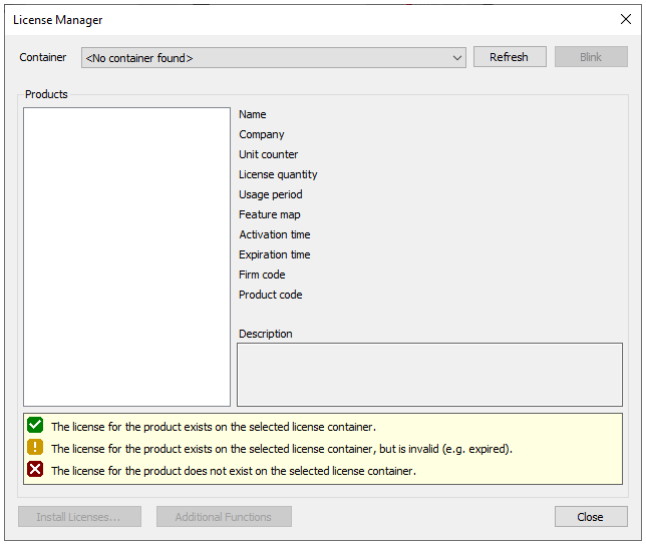

Figure 207: License Manager Dialog

- *Products*: A list of all installed CODESYS products subject to licensing is displayed, showing those found on the computer or device. If licenses are provided via containers, the containers appear as nodes in the license list. A preceding symbol indicates whether the license is present on the computer and valid. On the right side of the window, the following details are shown for the selected product and its corresponding licenses: *Name*, *Company*, *Unit counter*, *License quantity*, *Usage period*, *Feature map*, *Activation time*, *Expiration time*, *Firm code*, *Product code*, *Description*.
- *Install Licenses*: Opens the *Install Licenses on <computer> – Select Operation* dialog.
- *Additional Functions*: Opens the menu with the functions *Wink*, *Return License*, *Restore License* and *Update Container*.

7.24.6. Start PACTware

The command launches the current version of PACTware if it is already installed on the computer. If the software is not installed, an error message is displayed to the user.

Note: The PACTware version needs to be 5.0.

7.24.7. Options...

7.24.7.1. CFC Editor

Symbol:

This sub-dialog of the *Options* dialog provides settings concerning the editing in a CFC (Continuous Function Chart) editor.

The *CFC Editor* is has four sub-dialogs: *General*, *View*, *Print* and *Dialog*.

Description of *General*:

- *Enable AutoConnect*: If this option is activated the following is true: When you drop CFC elements somewhere on the canvas, unconnected pins that are touching each other will be connected automatically. This might be useful for quick editing, however take care not to create connections accidentally when moving elements around.

Description of *View*:

- *Display grid points*: Display all the points that are possible to have a connection on the background grid.
- *Show box icon*: Show the icons that boxes have by default. Like *AND* box.
- *Edit Line Colors...*: Opens the *Edit Line Colors* dialog, where it's possible to edit the line color of each data type.
- *Font (click on the sample to edit)*: When clicking in the sample, it's possible to select a *Font*, *Font Style* (like bold or italic) and *Size*. Also, there's the *Script*, that can select other type of alphabet.

Description of *Print*:

- *Fit method*: List box for fitting
- *Scale*: It's where the page scale can be set.

Description of *Monitoring*:

- *Number of displayed digits*: Number of displayed digits for floating-point numbers in the monitoring field.
- *String length*: Maximum length of string variable values in the monitoring field.

7.24.7.2. Debugging

Symbol:

This option in the dialog determines whether breakpoints remain enabled after a reset.

- *Restore Breakpoints After Reset*: If the checkbox is enabled (it's enabled by default), the *Reset Cold* or *Reset Warm* command will keep all set breakpoints active. Otherwise, these commands will disable all breakpoints.

7.24.7.3. Declaration Editor

Symbol:

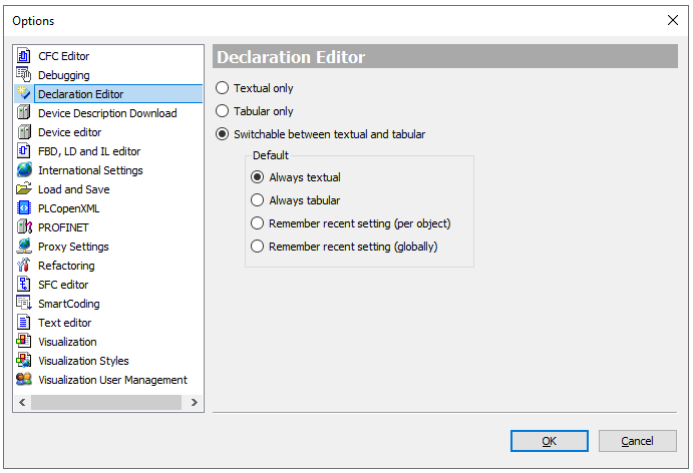

Figure 208: Declaration Editor

Define, which views of the declaration part should be available when an object is opened for editing:

- *Textual Only*: If this setting is activated the declaration editors will be displayed in a textual view.
- *Tabular Only*: If this setting is activated the declaration editors will be displayed in a tabular view.
- *Switchable between textual and tabular*: If this setting is activated, two buttons will be available in the declaration editor window for switching between the textual and the tabular view of the declaration part. In this case choose also one of the following settings, which defines in which of the both formats by Default the declaration editor at first will appear, when an object is opened for editing: *Always textual*, *Always tabular*, *Remember recent setting per object* (when reopening an object, the declaration editor will be opened in the same format as it had for this object during last editing) or *Remember recent setting globally* (when opening any object, the declaration editor format recently used in any object will be used).

7.24.7.4. Device Description Download

Symbol: 

The dialog is used to configure the addresses of download servers for device descriptions. By default, https://store.codesys.com/CODES is entered as the download server.

When you click the *Download Missing Device Descriptions* button in the *Device Repository* dialog, MasterTool IEC XE uses the servers specified here along with the configured proxy server credentials to download the necessary device descriptions.

- *Enter new download server here*: When double-click the object, an input field appears, allowing the user to specify the server's URL address.
- *Del*: Removes the selected download server.

7.24.7.5. Device Editor

Symbol: 

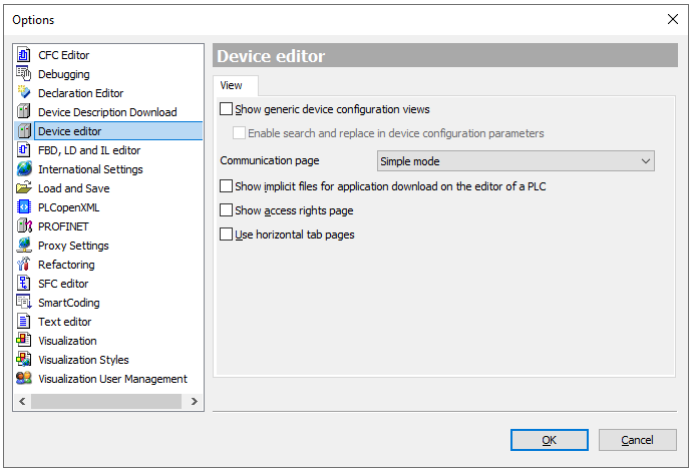

Figure 209: Device Editor

The dialog contains settings for configuring the display of the device editor.

There's a better description of the dialog:

- *Show generic device configuration views*: If checked, the tab displaying the list of device parameters is accessible in the device editors of parameterizable devices.
- *Communication page*: Sets the mode of the between *Classic mode* and *Simple mode*. The default is *Simple mode*.
  - *Classic mode*: The Communication tab of the device editor is displayed as a split window. The left side shows the currently configured gateway channels in a tree structure, while the right side displays the associated data and information.
  - *Simple mode*: The Communication tab is displayed as described in the relevant chapter of the help documentation.
- *Show implicit files for application download on the editor of a PLC*: If checked, the *Synchronized Files* tab becomes available in the device editors. Synchronized files are downloaded to the PLC during the application download process. These files can include external files added to the application or implicit files such as a source code archive.
- *Show access rights page*: If selected, the Access Rights tab will be available in the device editors.
- *Use horizontal tab pages*: The tabs will be displayed in horizontal (top of the screen), not in vertical.

7.24.7.6. FBD, LD and IL Editor

Symbol: 

This dialog of *Options* enables user to perform settings for editing in editors FBD, LD and IL.

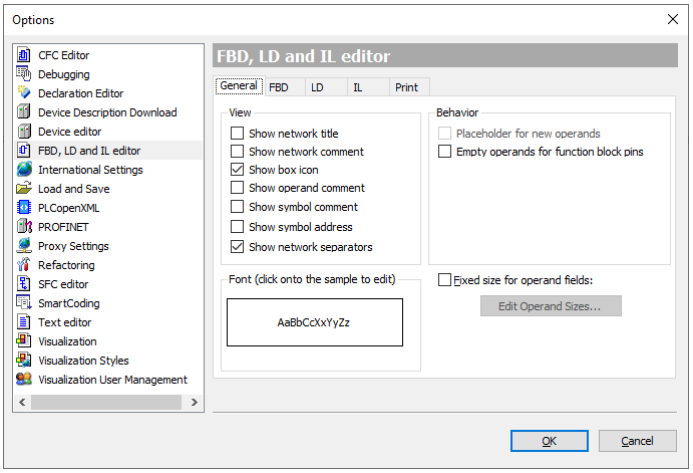

Figure 210: FBD, LD, and IL editor, *General* tab

View block:

General view:

- *Show network title*: the title of the network - if set - is displayed in the upper-left corner of a network.
- Show network comment: the comment - if set - network is displayed in the upper-left corner of a network. If the title of the network is visible, the comment will appear on the line below the title.
- Show box icon: If a function block or variable available in the library or in the properties of the object are provided with an icon (bitmap), this will be displayed inside the box in the FBD and LD editors. The standard operators also include icons.
- *Show operand comment*: the comment - that can be assigned to a variable on the implementation part of the editor - is displayed. The comment of the operand only refers to the current position of variable use (unlike *symbol comment* , which is defined in the declaration of a variable).
- *Show symbol comment*: the comment of each symbol (variable) will be displayed above the identifier. Can optionally be assigned a local *operand comment* .
- *Show symbol address*: for each symbol (variable) the assigned address is displayed above the symbol identifier.
- *Show network separators*: A separator is shown between the individual networks.

Font block:

At clicking in the sample, it's possible to edit the font, font style, font size, etc.

Behavior block:

- *Placeholder for new operands*: MasterTool IEC XE don't support this feature.
- *Empty operands for function block pins*: With this option enabled empty operands are allowed for pins of function blocks.

- Fixed size for operand fields: Available only to display comment in single-line network. In this option, the following parameters determine the size of fields of information assigned to an operand:
  - Width of the operand, characters: maximum number of characters of the name of an operand that are displayed.
  - Height of the operand, lines: maximum number of rows displayed to the name of an operand.
  - Height of the comment of the operand, lines: maximum number of lines available for the comment of an operand.
  - Height of the comment symbol, lines: maximum number of lines available for the comment of the symbol of the operand.

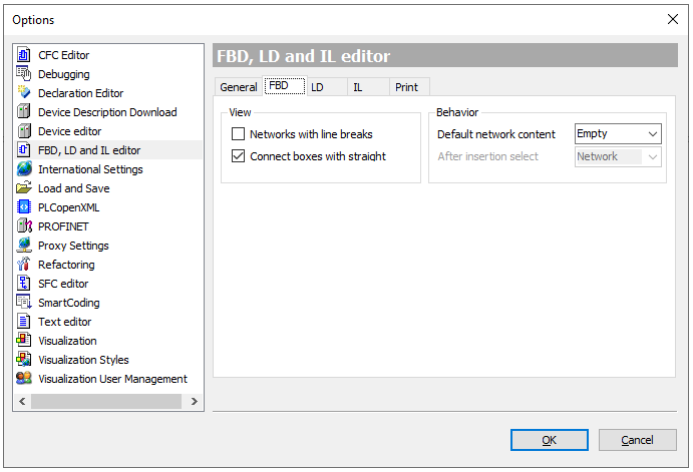

Figure 211: FBD, LD, and IL editor, *FBD* tab

The *FBD*, *LD* and *IL* tabs are composed of *View* and *Behavior* blocks. In each tab containing little differences of what is contained in those blocks.

View block:

- *Networks with linebreaks*: When enabled, a line of network connections will be broken in a way that the elements are visible, as much as possible on the current width of the window. This can cause the networks to be enlarged in height. If the editor window is too small the network will not be broken. This option is only available for *FBD* and *LD* tabs.
- *Connect boxes with straight line*: In this option, the network components are arranged so that their lines are short and fixed. In this way, the horizontal space required for the display of networks is reduced as far as possible. This can affect the highest boxes, for example, to provide enough space for input and output. If this option is disabled, the elements will keep its default size and connection lines (adapt space). This option is only available for the *FBD* tab.
- *Enable IL*: The IL implementation language is accessible within the development system. This option is only available in the *IL* tab.

Behavior block:

- *Default network content*: Defines what content is displayed in the editor when inserting a new network (empty, assigned or empty box).
- *After insertion select*: This option defines which element is selected after a new network is entered (Network or Element). This option is disable in the *FBD* and *LD* tabs.

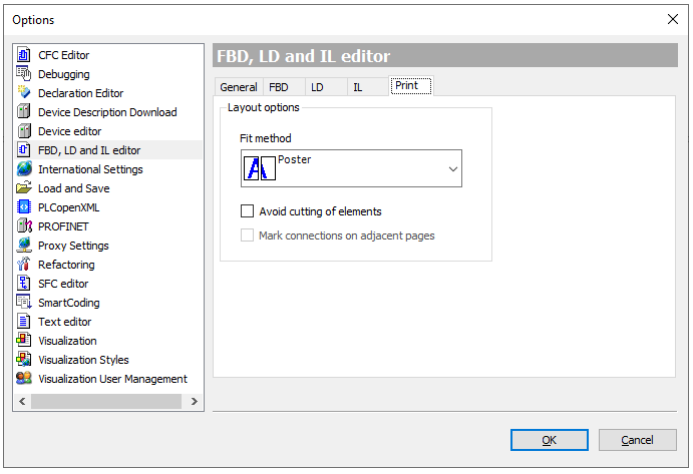

Figure 212: FBD, LD, and IL editor, *Print* tab

There's also the *Print* tab, where the user can find options to configure the layout options for the *Print...* command.

- *Fit method*: List box for fitting
- *Avoid cutting of elements*: Elements which do not fit on the page are printed on the next page.
- *Mark connections on adjacent pages*: Only can be selected when *Avoid cutting of elements* is enabled.

7.24.7.7. International Settings

Symbol: 

This sub-dialog of the Options dialog allows the following settings concerning the language used in the user interface and help system.

7.24.7.7.1. User Interface Language

- *Same as Microsoft Windows*: If this option is activated, the language used in the user interface will be that which is currently set by Microsoft Windows on your computer.
- *Specific culture*: If this option is activated, the language currently selected from the list will be used.

Note: Changing the user interface language will not be effective until MasterTool IEC XE is restarted. Some components may not be available in the selected language and will appear in their default culture, which typically is English.

7.24.7.7.2. Online Help Language

- *Same as user interface language*: This option is set by default. The online help will be displayed in the language, which is set for the user interface. If no help is available in this language, English will be used.
- *Specific culture*: If this option is activated, the language currently selected from the list will be used.

7.24.7.8. Load and Save

Symbol: 

This sub-dialog of the Options dialog allows settings concerning the behavior of MasterTool IEC XE at loading and saving a project.

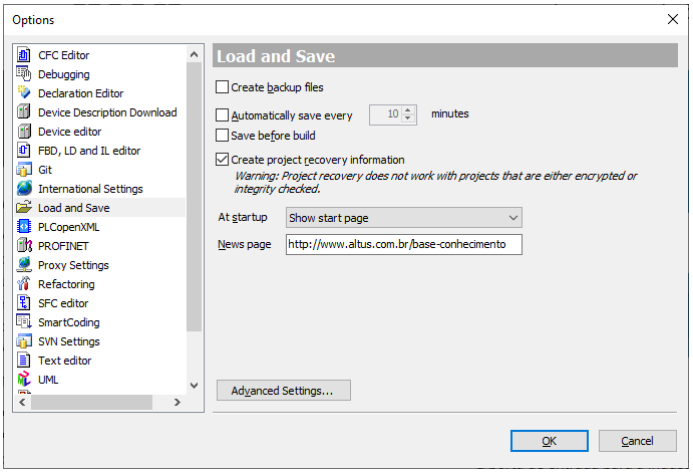

Figure 213: Load and Save

- *Create backup files*: If this option is activated, at each saving the project will not only be saved in <projectname>.project. but also copied to a file <projectname>.backup. If needed you can rename this backup-file and re-open in MasterTool IEC XE
- *Automatically save every ... minutes*: If this option is activated, the project will be automatically saved in the defined time intervals to a file <projectname>.autosave in the projects directory. After a non-regular termination of the programming system you might reload this autosave-file.
  If you regularly save or close the currently opened file, the associated auto save file will be deleted, in case of an irregular abortion it will be kept.
  If you reopen a project for which an appropriate auto save file is found, an *Auto Save Backup* dialog will appear, where you can decide whether to reopen the auto save project or the last version, which was saved by the user.
- *Save before build*: MasterTool IEC XE automatically saves the project before each build operation.

- *Create project recovery information*: If a project crashes during editing, a prompt will appear the next time the project is opened, asking whether you want to restore the unsaved data and create a new project file. If you click *Yes*, another dialog opens, allowing you to choose between opening the restored project or viewing a project comparison. This comparison highlights the differences between the last saved version and the restored project.
- On *At startup* option is defined what should happen when MasterTool IEC XE gets started:
  - Load last loaded project: The project, which was edited last, will automatically be opened.
  - Show *Open Project* dialog box: Corresponds to command *Open Project*.
  - Show *New Project* dialog box: Corresponds to command *New Project*.
  - Show empty environment: The programming system will be started without opening or creating a project.
  - Show *Start Page*: The *Start Page* view will be displayed. Corresponds to command *Start Page*.

7.24.7.9. PLCopenXML

Symbol: 

The dialog provides settings for how MasterTool IEC XE handles PLCopenXML export and import.

The *PLCopenXML export settings* has the following options:

- *Additionally export declarations as plain text*: When this option is selected, both formatting and comments are preserved. MasterTool IEC XE exports the plain text of the declaration part to the PLCopenXML file, thereby extending the PLCopenXML schema. *Export folder structure*: When this option is selected, MasterTool IEC XE also exports folders that contain any of the selected objects.

The *PLCopenXML Import Settings* has the following options:

- *Import folder structure*: When this option is selected, if the import file contains information about the folder structure of the object, MasterTool IEC XE imports the structure along with the objects. Otherwise, MasterTool IEC XE imports the objects without their folder structure.

7.24.7.10. Proxy Settings

Symbol: 

This dialog is used to store authentication data for the proxy server that MasterTool IEC XE uses to access the Internet. It is required when Internet access is routed through a proxy server within the network.

- *Use proxy settings*: Enable or disable the use of the registered proxy settings.
- *Enter Proxy Credentials*: A double-click opens a dialog prompting for the proxy server's username and password. MasterTool IEC XE uses these credentials to establish connections to the download server for libraries and device descriptions, as well as for accessing the MasterTool IEC XE system and executing the *View* > *Start Page* command. This button is available if your computer or network accesses the Internet via a proxy server.

7.24.7.11. Refactoring

Symbol: 

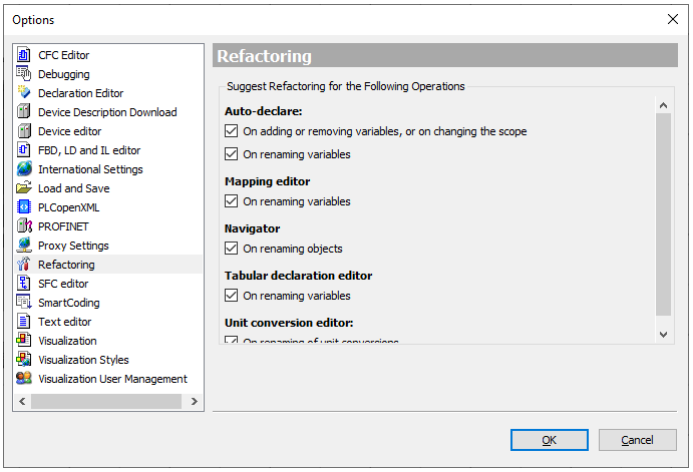

Figure 214: Refactoring

This dialog is used to define the project operations for which automatic refactoring is suggested, supporting your efforts to improve the code.

- *Auto-declare*: When you use AutoDeclare (Shift+F2) to rename a variable in its declaration, the 'Apply changes by refactoring' option is displayed. This opens the Refactoring dialog, where you can apply the variable name change across the entire project.
  - *On adding or removing variables, or on changing the scope*: If selected, delete the names in the *Declare Variable* dialog and click *OK* to close it. This will open the *Refactoring* dialog to remove the variable across the entire project.
  - *On renaming variables*: If selected, enter the names in the *Declare Variable* dialog and click *OK* to close it. This will then open the *Refactoring* dialog to rename the variable across the entire project.
- *Mapping editor*:
  - *On renaming variables*: If selected, when you change a variable name in the device editor (*I/O Mapping* tab), you will be prompted to confirm whether MasterTool IEC XE should perform *Automatic Refactoring* during the rename.
- *Navigator*:
  - *On renaming objects*: If selected, when you change the name of an object in the device tree or the POU view, you will be prompted to decide if MasterTool IEC XE should perform *Automatic Refactoring* during the rename.
- *Tabular declaration editor*:
  - *On renaming variables*: If selected, when you change the name of a variable in the tabular declaration editor, you will be prompted to confirm whether MasterTool IEC XE should perform *Automatic Refactoring* during the rename
- *Unit conversion editor*:
  - *Unit conversion editor*: If selected, when you change the name of a conversion in the unit conversion editor, you will be prompted to decide whether MasterTool IEC XE should perform 'Automatic Refactoring' during the rename.

7.24.7.12. SFC editor

Symbol: 

This sub-dialog of the *Options* dialog provides sub-dialogs for settings concerning the editing in a SFC (Sequential Function Chart) editor. The settings made in each of these dialogs will immediately be applied to the currently opened editor views as soon as they have been confirmed (and the dialog closed) with *OK*.

7.24.7.12.1. *Layout*

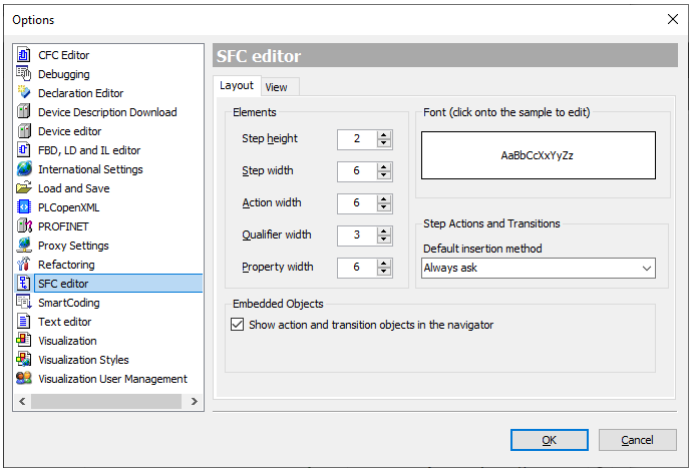

Figure 215: *Layout* tab

The settings are to be made in *grid units*. One grid unit is equal to font size currently set in the text editor options.

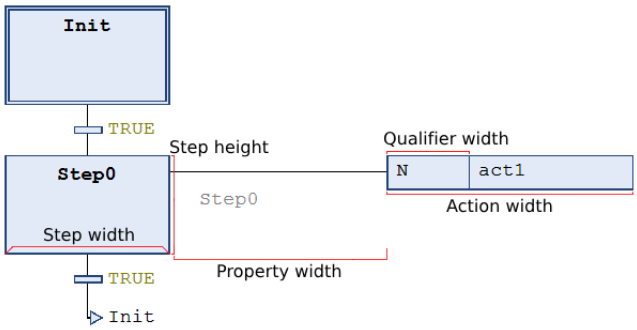

Figure 216: LayoutOptions

- *Step height*: Step element height in grid units. Possible values: 1-100.
- *Step width*: Step element width in pixels in grid units. Possible values: 2-100.
- *Action width*: Action element display width in grid units. Possible values: 2-100.
- *Qualifier width*: Qualifier display width in grid units. Possible values: 2-100.
- *Property width*: Property display width in grid units. Possible values: 2-100.

- *Font*: It's possible to change the font, font size, and font style.
- *Default insertion method*:
  - *Copy reference*: When a step is copied, the references to the action objects that call the step are also copied. Both the copied step and the new step will call the same action.
  - *Duplicate implementation*: The references to the action objects that call the step are linked to it. When the step element is copied, new action objects are created for the new step, and the implementation is duplicated.
  - *Always ask*: When inserting a step action, you are always prompted to choose whether the actions of a step element should be duplicated when copied or if the reference to the existing action should be applied.

    However, if a step already contains an embedded action, any newly inserted actions will also be embedded. Similarly, if the step contains a non-embedded action, new inserted actions will follow that mode. In these cases, you will no longer be prompted to select a duplication mode.
- *Show action and transition objects in the navigator*: Action and transition objects *embedded* in the SFC box with a step are displayed in the *Devices* tree view or the *POUs* tree view.

7.24.7.12.2. *View*

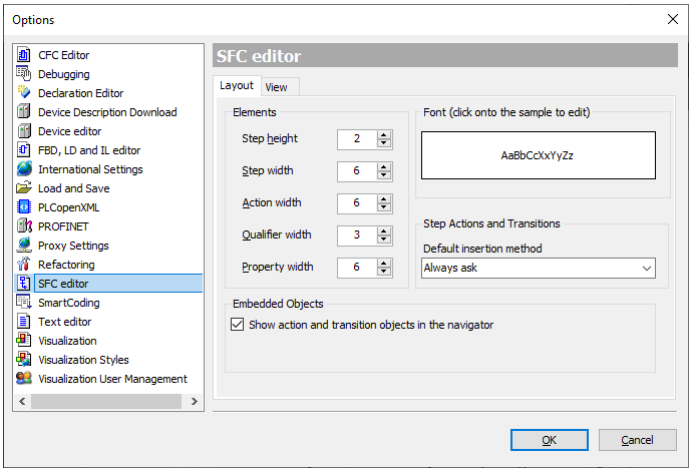

Figure 217: *View* tab

Tab divided in *Property Visibility* and *Online* .

7.24.7.12.3. *View - Property Visibility*

Here you can define which of the element Properties (step attributes) should be displayed next to an element in the SFC editor view: The *Common* and the *Specific* properties for each element type are shown in the table and you can each activate the checkbox in the *Value* field in order to get displayed the property value right to the element in each SFC object. If you additionally activate the checkbox in the *With Name* field, the value will be preceded by the name of the property.

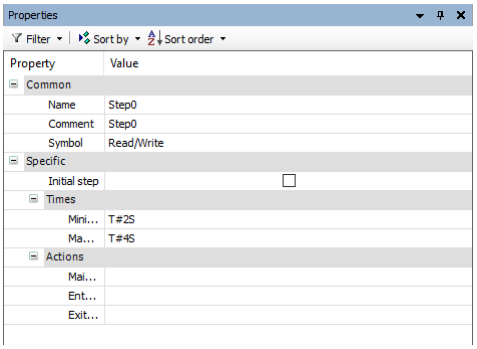

Figure 218: Properties

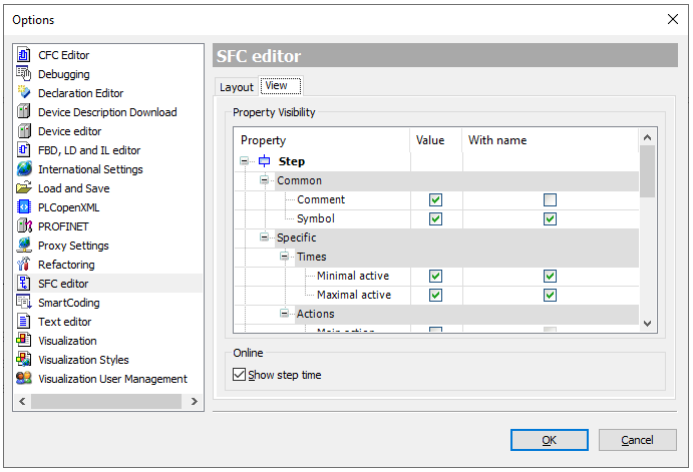

Figure 219: Editor *View*

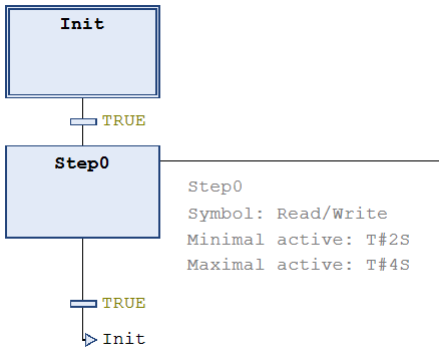

Figure 220: Property Visibility Example

7.24.7.12.4. View - Online

If option *Show step time* is activated, in online mode the current step time will be displayed right to each step element for which time properties are set.

7.24.7.13. SmartCoding

Symbol: 

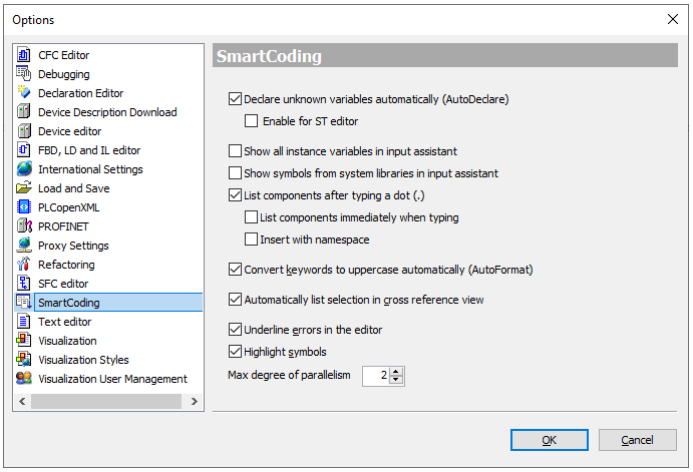

Figure 221: SmartCoding

This sub-dialog of the *Options* dialog allows some settings for making coding more pleasant. This concerns the Smart Coding functions like for example *AutoDeclaration* or *Input Assistant*.

- *Declare unknown variables automatically (AutoDeclare)*: If this option is activated, the Autodeclare dialog will open automatically when you enter a not yet declared identifier in any programming language editor.
  1. *Enable for ST editor*: The *Declare unknown variables automatically (AutoDeclare)* option must be selected for the *AutoDeclare* function to be available in the ST editor. If selected, the *AutoDeclare* feature can be used in the ST editor; otherwise, it will not be accessible.
- *Show all instance variables in input assistant*: If checked, the List Components function allows you to select the local variables of a function block instance. Otherwise, it only allows selection of the input and output variables of the function block instance.
- *Show symbols from system libraries in input assistant*: System libraries are automatically inserted into the *Library Manager* and displayed in light gray. If this option is checked, the *Input Assistant* will offer symbols such as global variables, data types, and function blocks. If not checked, the symbols from the system libraries will not be available in the *Input Assistant*.
- *List components after typing a dot (.)*: This means that when entering a dot (.) in an editor at a position where an identifier is expected, you will get a selection list with possible entries.
  - *List components immediately when typing*: As soon as you enter any character in the editor, a list will open which contains all available identifiers and operators. Depending on which character sequence you have entered, automatically the first entry of this list matching this character sequence will be selected. Anyway, you can select any item in the list by placing the cursor on the required item and pressing <ENTER> key.
  - *Insert with namespace*: MasterTool IEC XE prefixes the identifier with the namespace.
- *Convert keywords to uppercase automatically (AutoFormat)*: If this option is activated, all keywords used in text parts of editors automatically will be written in capital letters. Example: If you enter *bVar:bool;* this will be converted to *bVar:BOOL;*.
- *Automatically list selection in cross reference view*: If this option is activated, the Cross Reference View always automatically lists the references of the variable currently selected in the active editor.
- *Underline errors in the editor*: If checked, incorrect or unknown program code will be underlined.
- *Highlight symbols*: If checked, all occurrences of a symbol at the cursor's position will be highlighted in color within the editor, making it easy to quickly identify cross-references.
- *Max degree of parallelism*: A list box for selecting the number of parallel threads that can be used for precompile processing is provided. MasterTool IEC XE automatically detects the number of threads based on the number of CPU cores, and this default value should only be changed in exceptional cases.

7.24.7.14. Text Editor

This sub-dialog of the Options dialog allows settings concerning the editing in a text editor.

7.24.7.14.1. *Theme*

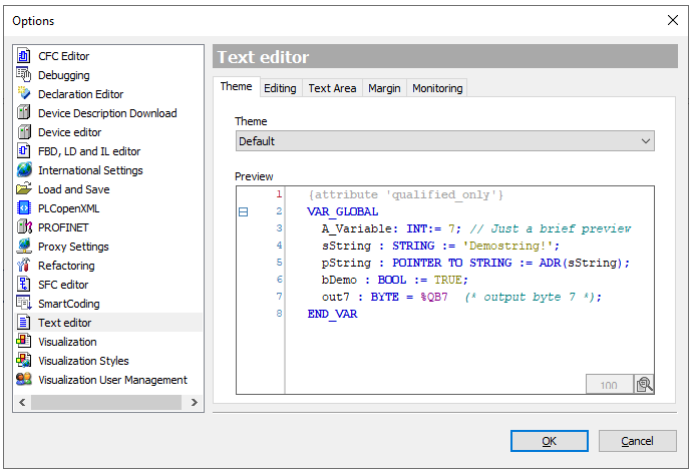

Figure 222: *Theme* tab

In this tab, it's possible to change the theme of the text editor.
There are two themes: *Default* (light) and *Dark*.

7.24.7.14.2. *Editing*

Definition of undo, tabulator, indenting, folding etc. in the editing area of the text editor.

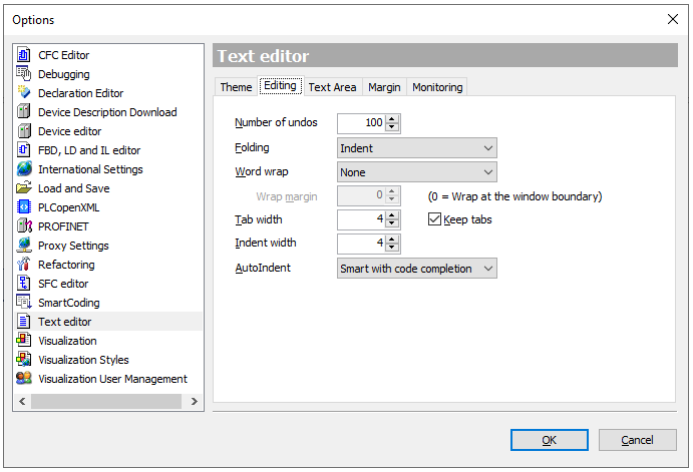

Figure 223: Editing

See in the following the description of the dialog:

- Number of undos: Define how many editing steps should be saved, so that they can be made undone by the Undo function.
- Folding: Define whether the code should be viewed structured by folds. This means that sections specified by indented position and marked by a special comment are hidden in a fold, which can be opened or closed via by a mouse-click on the plus- resp. minus-sign on the left side of the folds header line. The options:
  - *None*: No folding is done
  - *Indent*: All lines indented to the previous one will be put in a fold, which is headed by the preceding line. By example, the lines between VAR and END_VAR and between IF and END_IF are indented, thus are packed in a fold indicated by a minus-sign in open and by the plus-sign in closed state).

- *Explicit*: The code section to be packed in a fold must be marked explicitly by comment lines: Before the section in a comment enter three opening braces , *after the section in a comment enter three closing braces* . The comment may contain further text also.

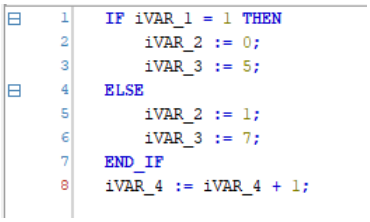

Figure 224: Opened Folds

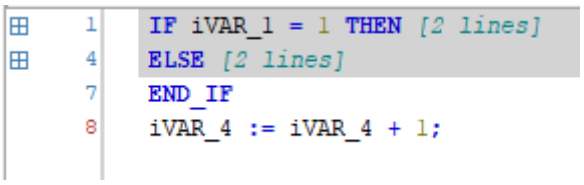

Figure 225: Closed Folds

Examples for defining a folding section with the *Explicit* option:

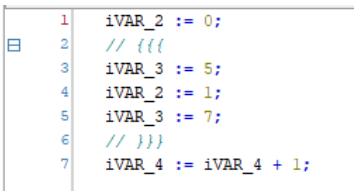

Figure 226: *Explicit* Example

- *Word Wrap*:
  - *None*: The line can be filled endlessly.
  - *Soft*: The lines will be broken at the boundary of the window.
  - *Hard*: The lines will be broken after the number of characters defined by Wrap margin. (A value of 0 means that the lines are wrapped at the window boundary).
- *Tab width*: Tab width in number of characters.
- *Indent width*: Indent width used in case of AutoIndenting that is the number of spaces at the beginning of a line.
- *Autoindent*: With the following options:
  - *None*: No automatic indenting. Editing always starts at the left margin of the editor's text area.
  - *Smart*: Code lines following a keyword (for example VAR or IF) are automatically indented according to the Indent width defined above.
  - *Smart with code completion*: When you enter a keyword like for example VAR or IF automatically the associated code lines following below are indented according to the Indent width defined above and additionally the terminating keyword is added (for example END_VAR or END_IF).
- *Keep Tabs*: If activated, a space which has been entered using the tab key according to the defined Tab width (see above) will not be resolved to single character spaces afterwards. So if the tab width gets changed later, the existing tab spaces will be corrected correspondingly.

7.24.7.14.3. Text Area

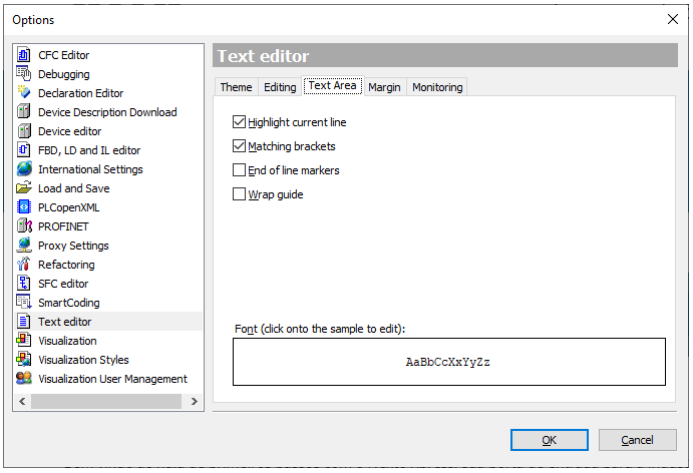

Figure 227: Text Area

- *Highlight current line*: The line where the cursor is located is highlighted.
- *Matching brackets*: When the cursor is placed before or after a bracket in a line of code, the matching opening or closing bracket is highlighted with a frame.
- *End of line markers*: A small dash marks the end of each editor line, appearing after the last character, including any spaces.
- *Wrap guide*: When a soft or hard line break is selected, a vertical line indicates the defined break position.
- *Font*: Clicking the field it's possible to change the font, font style, font size, etc.

7.24.7.14.4. Margin

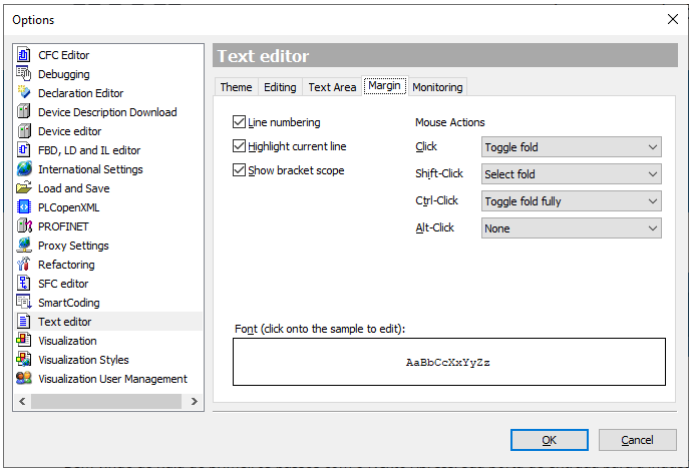

Figure 228: TextEditorMargin

This tab are some settings for the left margin of the text editor window, separated from the input area by a vertical line.

- *Line numbering*: The declaration and implementation sections of the editor are numbered on the left, with each starting at 1.
- *Highlight current line*: The line number of the current cursor position is highlighted.
- *Show bracket scope*: Brackets encompass the lines between the opening and closing keywords of a construct (e.g., IF and END_IF). When this option is enabled, and the cursor is placed before, after, or within one of the keywords, a square bracket appears in the margin to indicate the bracketed area.

Mouse Actions: You can assign specific actions to mouse actions or mouse/keyboard combinations. MasterTool IEC XE performs the selected action when you move the mouse to the plus or minus sign in front of a bracketed area's header. There's the possible options that can be selected:

- *None*: The mouse actions does nothing.
- *Select fold*: MasterTool IEC XE selects all lines within the bracketed area.
- *Toggle fold*: MasterTool IEC XE opens or closes the bracketed area, and if there are nested brackets, it toggles the first level of the area.
- *Toggle fold fully*: MasterTool IEC XE opens or closes all levels of a nested bracketed area.

The *Font* field, by cliking at it, it's possible to change the font, font size, font style, etc.

7.24.7.14.5. Monitoring

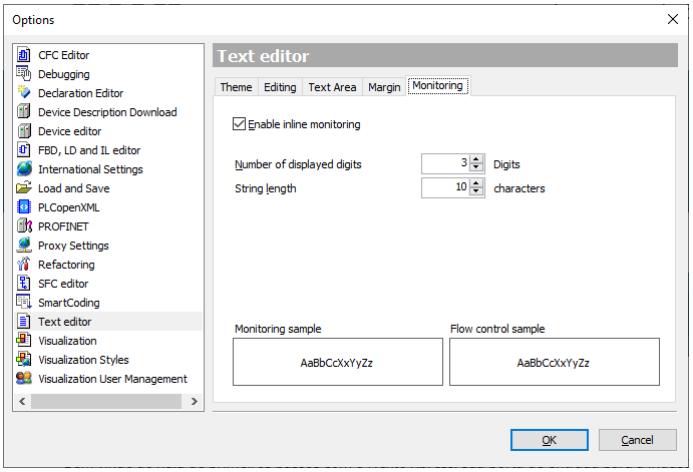

Figure 229: Monitoring

- *Enable inline monitoring*: Display of monitoring fields after the variables in online mode.
- *Number of displayed digits*: Number of displayed digits for floating-point numbers in the monitoring field.
- *String length*: Maximum length of string variable values in the monitoring field.

7.24.7.15. Visualization

There are the options where you can configure for the *Visualization* object.

7.24.7.15.1. General

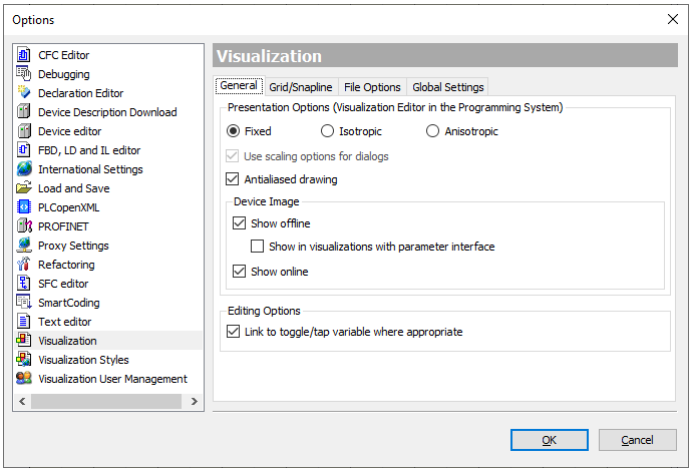

Figure 230: *General* tab

Presentation Options (Visualization Editor in the Programming System):

- *Fixed*: The visualization retains its original size.
- *Isotropic*: The visualization retains its proportions.
- *Anisotropic*: The visualization adjusts dynamically to fit the window in the development environment.
- *Use scaling options for dialogs*: The dialogs, including those for the keypad and numpad, are scaled in the same way as the visualization, using the same scaling factor. This ensures that if a dialog is designed to match the visualization, they will scale together seamlessly.
- *Antialiased drawing*: Antialiasing methods are applied to the visualization, both during editing and when integrated into the runtime display.

Device Image:

If a *device image* (DeviceImage) is defined in the controller's device description or in the Advanced Settings of the Visualization Manager, it can be displayed in the visualization editor.

- *Show offline*: The device image is visible in offline mode.
- *Show in visualizations with parameter interface*: Display in visualizations with a parameter interface
- *Show online*: The device image is visible in online mode.

Editing Options:

- *Link to toggle/tap variable where appropriate*: The <*toggle/tap variable*> placeholder is activated in the visualization element properties. When an element with *Color* variables and *Toggle* color properties is dragged into the visualization editor, this property is automatically configured with the <*toggle/tap variable*> placeholder. This applies to the following elements: Button, Frame, Image, Line, Pie, Polygon, Rectangle, Text Field, and Scroll Bar.

7.24.7.15.2. Grid/Snapline

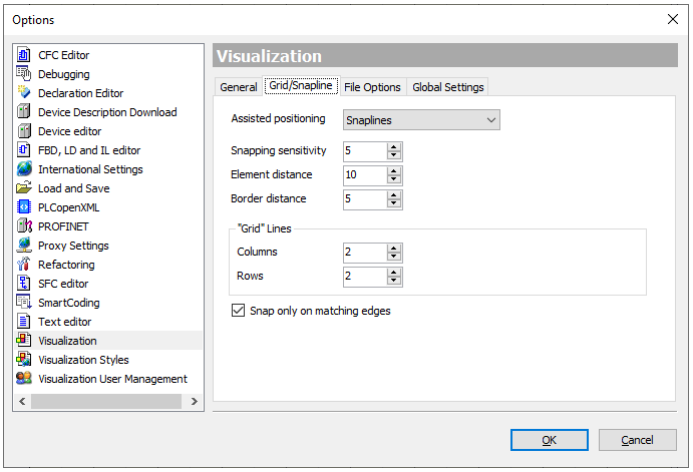

Figure 231: *GridSnapline* tab

- *Assisted positioning*: The Snaplines option is selected by default in the editor. When you insert and move elements, lines appear to indicate positions that align with adjacent elements. Additionally, the visualization editor is divided into a grid, which further assists in aligning elements with one another or arranging them in a fixed column and row layout.
- *Snapping sensitivity*: The distance in pixels at which snapping to the snapline becomes active.
- *Element distance*: Minimum distance to adjacent elements *Border distance*: The minimum distance from the border that is taken into account by the snapline.

Grid lines:

- *Columns*: The number of columns, all of equal width.
- *Rows*: Number of rows (all the same width)
- *Snap only on matching edges*: The alignment is restricted; it only occurs between edges that share the same position, such as aligning the left edge with another left edge.

7.24.7.15.3. File Options

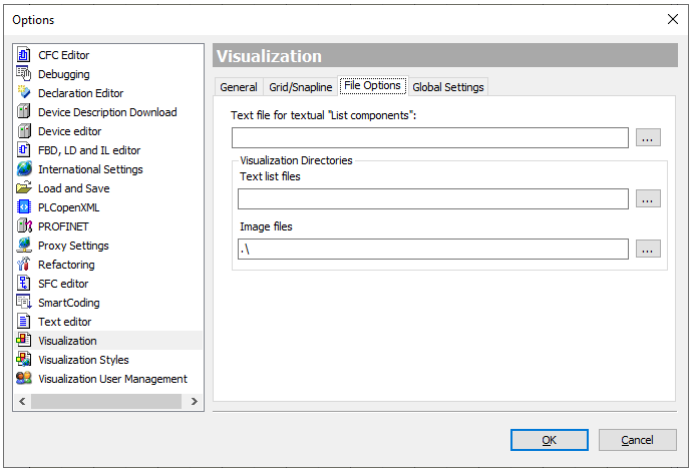

Figure 232: *FileOptions* tab

- *Text file for textual List components*: The file name and location pertain to a .csv file type that contains a table formatted as a *Text List*. The entries in this file are generated when the *List components* feature is utilized for input assistance. Note that this file is created as an export of the global text list through the *Import/Export Text Lists* command.

- *Text list files*: Locations for Text Lists.
- *Image files*: The directory contains image files available in the project, with multiple directories separated by a semicolon. MasterTool IEC XE utilizes this directory for image pools that include image files not integrated into the project but referenced by a link. Absolute paths are used for global images across the entire project, while relative paths (for example, .\images) are employed for project-specific images. This option ensures that image pools can be opened successfully even when the project and its associated image directory are relocated to another location on the computer.

7.24.7.15.4. Global Settings

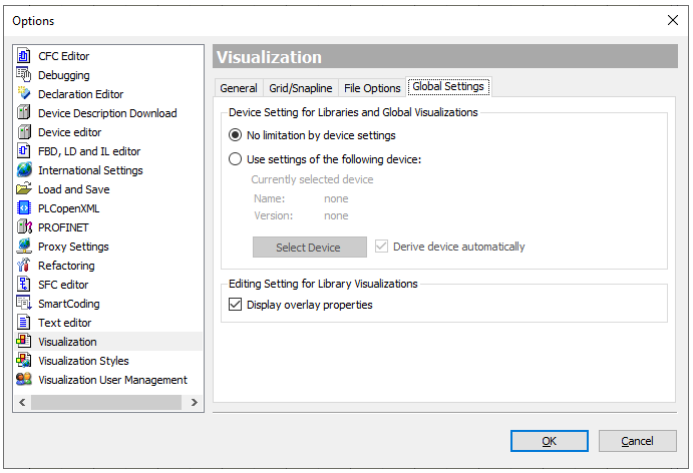

Figure 233: *GlobalSettings* tab

Device Setting for Libraries and Global Visualizations block:

- *No limitation by device settings*: All properties of visualization elements are available for constructing the visualization.
- *Use settings of the following device*: Only the properties of visualization elements released by the device settings below are available for constructing the visualization.
  - *Select device*: Opens *Select Device* dialog, with all installed devices.
  - *Derive device automatically*: If a device is present in the project, its device settings are applied, which means that only the visualization element properties that the device can implement are available. For instance, a gradient or a rotating element (interior rotation property) may not be supported by all devices.

Editing Setting for Library Visualization block:

- *Display overlay properties*: The properties related to the overlay feature are shown.

7.24.7.16. Visualization Styles

Symbol: 

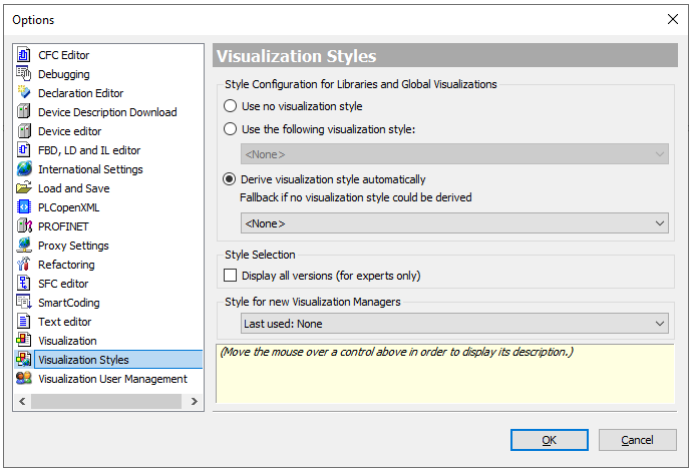

Figure 234: Visualization Styles

The dialog is utilized to configure the display of library visualizations and visualizations in the POU view within the visualization editor. Additionally, it serves to configure the *Visualization Manager* tab under the *Settings* (*Style Settings* group). It's important to note that these settings are not applied during visualization runtime; at that time, only the settings from the *Visualization Manager* are accessible on the *Settings* tab.

- *Style Configuration for Libraries and Global Visualizations*:
  - *Use no visualization style*: If this option is checked, visualizations in the POUs view or from libraries will behave as though no style has been selected.
  - *Use the following visualization style*: If selected, visualizations in the POUs view or from libraries will behave as if the chosen style is active.
  - *Derive visualization style automatically*: If selected, visualizations in the POUs view or from libraries will derive their style configuration from the Visualization Managers of the current project. If this is not possible or the result is ambiguous, the style chosen below will be used as a fallback solution.
  - *Fallback if no visualization style could be derived*: The style selected from the list box serves as a fallback solution if no visualization style can be derived.
- *Style selection*: The selected style list box can be configured in the *Visualization Manager* under the *Settings* tab, within the *Style Settings* group.
  - *Display all versions (for experts only*: If not checked, all other styles from the repository, including the selected style, will be listed for selection, but only in their latest version. If newer versions of the selected style are installed, they will also be listed. Otherwise, all installed styles across all available versions will be selectable.
- *Style for New Visualization Managers*:
  - *Last used: <style, version, vendor>*: The style that is automatically selected when adding a new visualization application may vary in display depending on the device, despite this setting.
  - *Preset: <style, version, vendor>*: A list box containing styles from the style repository.

7.24.7.17. Visualization User Management

Symbol: 

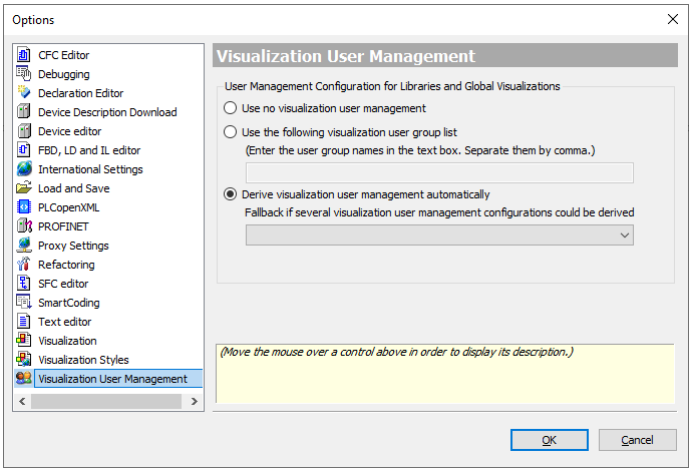

Figure 235: Visualization User Management

User management configuration for libraries and global visualizations: the option is used for visualizations within libraries and for global visualizations located under the POUs tree.

- *Do not use visualization user management*: tick this option then the affected visualizations will behave as if no user management configuration is assigned.
- *Use the following user group list for visualization*: tick this option then the affected visualizations will use the groups listed in the text field. It can be edited manually or by using the button Export groups for global visualization in dialog User Management.
- *Derive visualization user management automatically*: tick this option then the affected visualizations will derive the user management configuration from the selected visualization manager. The selection list provides all actually installed Visualization Managers of the project. If this is not possible then the user group list from option Use the following visualization user group list will be used.

The user management configuration of a visualizations under the device tree have to be done in User Management of the associated visualization manager.

7.24.8. Miscellaneous

Sub-menu for commands that does not fit in other menus.

- *Generate PROFINET Device from GSDML*: The command opens the *Generate PROFINET Device from GSDML* dialog, allowing MasterTool IEC XE to generate devdesc.xml files from a custom-created GSDML file for a PROFINET device.

Figure 236: Generate PROFINET Device from GSDML

- *Source GSDML*: GSDML file used to generate the device description for a PROFINET device.
- *DAP*: Data access point within the GSDML file
- *Devdesc ID*: Unique ID for the device description in the device repository to be generated. The suggested value can be modified.
- *Devdesc version*: Version for the device description in the device repository to be generated. The suggested value can be overwritten.

7.24.9. Scripting

Sub-menu find in the menu *Tools* to help the use run Python scripts. In MasterTool IEC XE there are two commands:

- *Execute Script File*: The command opens a dialog to select and run a script file (*.py).
- *Enable Script Tracing*: The command instructs MasterTool IEC XE to print all commands from the script file to the message view. Use this command to monitor and debug scripts. A blue border around the symbol indicates that the option is enabled.

7.25. Window Menu

The frame provides commands for command category Window that can be used to handle the user interface windows.

- Next Editor
- Previous Editor
- Close All Editors
- Reset Window Layout
- New Horizontal Tab Group
- New Vertical Tab Group
- Float
- Dock
- Auto Hide
- Next Pane
- Previous Pane
- Window <n>
- Windows..

7.25.1. Next Editor

Default Shortcut: <CTRL>+<F6>

If several editor windows are currently opened, this command can be used to change the focus from the current window to the nest one. This is the window represented with the tab to the right of the currently active window.

7.25.2. Previous Editor

Default Shortcut: <CTRL>+<SHIFT>+<F6>

If several editor windows are currently opened, this command can be used to change the focus from the current window to the previous one. This is the window represented with the tab to the left of the currently active window.

7.25.3. Close Editor

Default Shortcut: <CTRL>+<F4>

This command closes the currently active editor window (not applicable to the *Configuration (Bus)*, except for Express PLCs).

7.25.4. Close All Editors

Symbol: 

This command closes all currently opened editor windows (not applicable to the Configuration (Bus), except for Express PLCs).

7.25.5. Reset Window Layout

This command serves to reset all currently open views to their default docking positions. A confirmation prompt will be displayed before the reset gets executed.

7.25.6. New Horizontal Tab Group

Symbol: 

This command is available, if several editor windows are arranged in tabs.

It will place the currently active tab window in a new, separate tab group below the existing one *active* window means that window, where the cursor was last placed. If you open a further object, that is a further editor window, this will be added to that tab group which contains the currently active window.

7.25.7. New Vertical Tab Group

Symbol: 

This command is available, if several editor windows are arranged in tabs.

It will place the currently active tab window in a new, separate tab group to the right of the existing one. *active* window means that window, where the cursor was last placed. If you open a further object, that is a further editor window, this will be added to that tab group which contains the currently active window.

7.25.8. Float

This command can be used to un-dock a window that is currently docked, that is fix part of the MasterTool IEC XE user interface frame. The window will become *floating* and can be positioned anywhere on the screen. To dock a floating window, use the *Dock* command.

7.25.9. Dock

This command can be used to dock a window, which previously has been undocked by a *Float* command.

7.25.10. Auto Hide

This command can be used to *hide* a window. The window will be represented by a tab at the border of the MasterTool IEC XE user interface and only be visible if you click with the cursor in this tab.

The command corresponds to the use of the Docking button in the upper right corner of a window.

7.25.11. Next Pane

Default Shortcut: <F6>

This command can be used in a window with two or several panes to get to the next pane.

Example: If in an object is opened in a ST editor window and the cursor is currently placed in the declaration part, with *Next Pane* the focus will change to the implementation part.

7.25.12. Previous Pane

Default Shortcut: <SHIFT>+<F6>

This command can be used in a window with two or several panes to get to the previous pane.

Example: If in an object is opened in a ST editor window and the cursor is currently placed in the implementation part, with *Previous Pane* the focus will change to the declaration part.

7.25.13. Window <n>

For each currently opened editor window the *Window* menu contains a command named <*object name*>, by which the window can be made active, that is the focus can be placed there. Behind the object name *(offline)* will be added for offline views, *(Impl)* or <*instance path*> will be added in case of function block views.

7.25.14. Windows...

This command opens the *Windows* dialog where you get listed all currently defined editor windows, that is windows used for editing any objects.

To activate a window, that is to set the focus to that window, select the respective entry and use button *Activate*.

7.26. Help Menu

The *Help* menu provides functions to get links and online help for MasterTool IEC XE users.

Available commands:

- Contents
- Index
- Search
- Contact Support
- Update Software License
- Documentation
  - Technical Specifications
  - Programming Manual
  - User Manual
- Release Notes
- Altus
- About

7.26.1. Contents

Symbol: 

Default Shortcut: <CTRL>+<SHIFT>+<F1>

This command opens the *Contents* tree of the online help in a separate window.

The contents is structured via *books* which can be opened or closed via a single mouse-click on the plus/minus-sign. The particular pages can be displayed in a separate window by a single mouse-click on the page entry.

7.26.2. Index

Symbol: 

Default Shortcut: <CTRL>+<SHIFT>+<F2>

This command opens a dialog showing a list of all index keywords provided by the online help in a separate window.

When you enter a string in the Look For field the first keyword matching this string will be selected in the list. By a double-click on list entry either the respective help page will be opened, or - in case of multiple pages found for this keyword, a list of help pages will be displayed in the Index Results window.

7.26.3. Search

Symbol: 

This command opens a search dialog in a separate window. This dialog supports a full-text search over all pages of the online help.

You can enter the string to be searched in the *Look For*: field.

The following search options can be set:

- *Search in titles only*: The string will only be searched in the titles of the help pages.
- *Display partial matches*: If the option is activated, the string will also be found if it is part of another string. Example: *log* will be also be detected in *dialog* whereas if the option is deactivated, *log* will only be detected if used in the help text as a separate string *log*.
- *Limit to ... matches*: The number of matches, which should be detected and reported in the Search result window, can be limited. The list of help pages containing the searched string will be displayed in the Search Results window, which will open automatically.

7.26.4. Contact Support

Symbol: 

This command opens the Altus web site, directly on the technical support page.

7.26.5. Update Software License

Symbol: 

This command opens a dialog box where you can change the company name, serial number and software key.

7.26.6. Documentation

It allows accessing the MasterTool IEC XE documentation.

Available commands:

- Technical Specifications
- Programming Manual
- User Manual

7.26.6.1. Technical Specifications

This command opens the default web browser to the Technical Specifications document on its most current version available on Altus website.

7.26.6.2. Programming Manual

This command opens the default web browser with the Programming Manual on its current version available on Altus website.

7.26.6.3. User Manual

This command opens the default web browser with the User Manual on its current version available on Altus website.

7.26.7. Release Notes

This command opens the *Release Notes* dialog, where it shows the *New features* and *Enhancements* from each MasterTool IEC XE version.

7.26.8. Altus Home Page...

This command opens the Altus website.

7.26.9. About

This command opens a box showing the version information on the currently used MasterTool IEC XE Programming System, the operational system, .NET, used components and the license data.

7.27. Trace

Provides commands for working with the *Trace* editor. These commands are available in the *Trace* menu when the editor is active.

ATTENTION

For comprehensive information and detailed guidance on using this feature, we recommend referring to the online help provided by CODESYS https://content.helpme-codesys.com/en/CODESYS%20Trace/_trace_menu_commands.html

8. Editors

8.1. General Regards on Editors

This chapter discusses the different types of editors available for configuration objects, devices, network settings, and other types of configurations. Each type of editor has its peculiarities. However, some features are general and apply to all editors.

Whenever there are numeric fields being set in editors, these fields have a minimum and maximum limit value, which depends on the functionality of the field. For example, a timeout field may vary in the value from 10 s (minimum) and 65535 s (maximum). In these cases it is not possible to configure values outside this range and this consistency is performed during Setup. For some numeric fields this consistency is performed in one of the stages of code generation. This also happens when there is a dependency between different fields, as for example between the cycle time of a task and its watchdog value.

In case of parameters representing %I, %Q and %M direct mappings addresses areas, the consistency is performed only the during code generation. As the projects are designed to any CPU models with different area sizes the fields representing the address ranges within these areas are limited between 0 and 2147483647. During the code generation the values assigned in the configuration are consisted with the limits to available each CPU model. In case the values are outside these limits an error message will be generated during the process.

When the *I/O Mappings* tab is opened and the *Reset Origin* command is run, it will not display show the mappings with its current values. In this case, it *Current Value* column displays <Bad>. Perform the *Download* command (Online menu) and close/open the module editor window so that the values can be properly displayed and updated. For further information see Reset Origin and Download.

8.2. Nexto Bus Editor

The bus editor feature already comes preset according to the CPU model and to the selected source in the wizard. Its configuration can be changed via the *Configuration (Bus)* option located in the device tree.

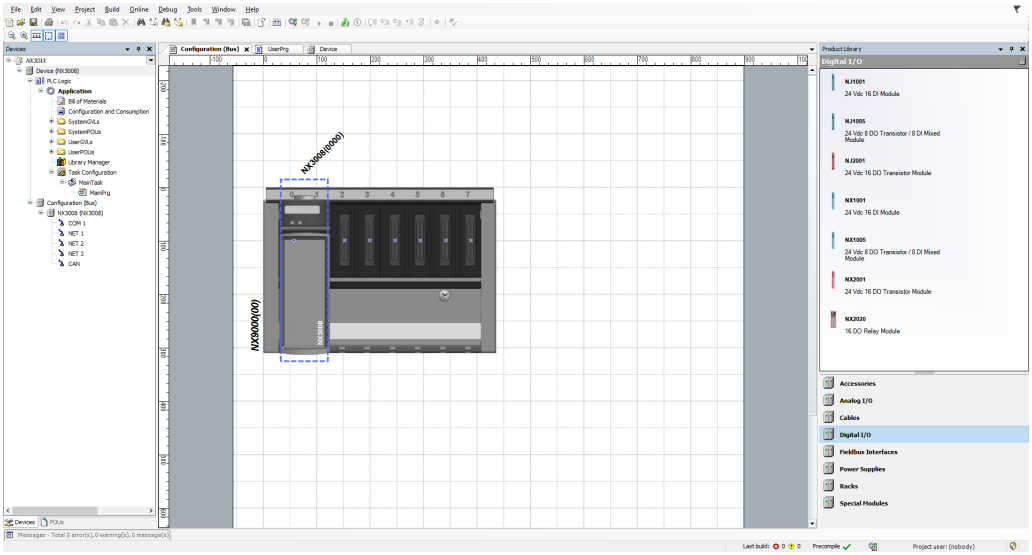

Figure 237: Bus Editor

8.2.1. Add Device

To add a device to the bus, it must be selected in the *Products Library* and dragged to the desired spot insertion.

If the added device is not properly connected to the bus, it will be marked with a rectangle and a diagonal red bar, as seen in the figure below:

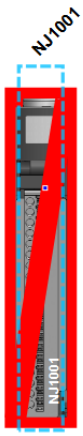

Figure 238: Disconnected Device

8.2.2. Remove Device

To remove a device on the bus, just click the right mouse button on it and select *Delete* or click the left mouse button on the device and click on <DELETE> key.

Note: Online changes cannot be applied when the devices parameters are changed on the bus or when devices are added or removed.

8.3. Nexto Digital I/O Module Editor

By adding a digital I/O module to the bus there are three possible configuration screens for it. They can be accessed by double-clicking the module.

8.3.1. Process Data

Process data are the variables used by the I/O module for access and control, as shown in the figure below:

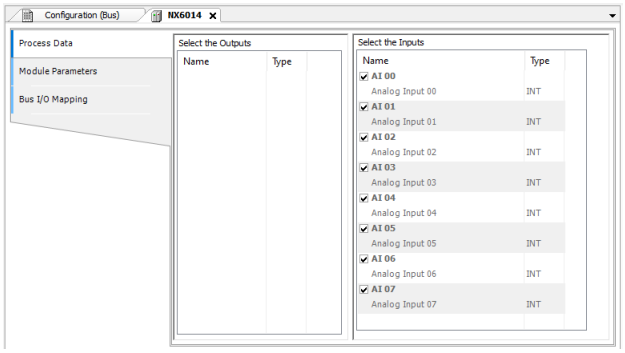

Figure 239: Disconnected Device

Information regarding each parameter is described in the current module Technical Characteristics.

8.3.2. Module Parameters

The modules parameters are configured through the screen shown in the image below:

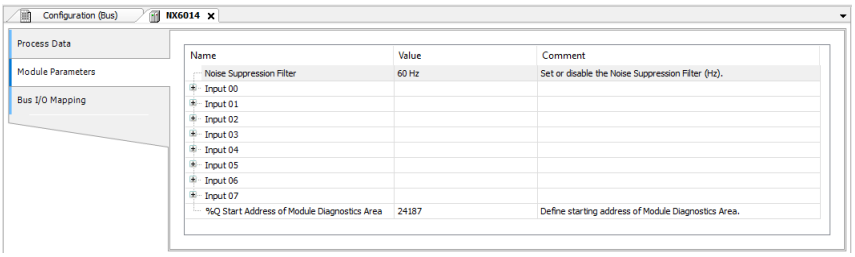

Figure 240: Module Parameters

The information about each parameters is described in the current module Technical Characteristics.

8.3.3. Bus: I/O Mapping

The fllowing figure shows the map of all the module inputs/outputs.

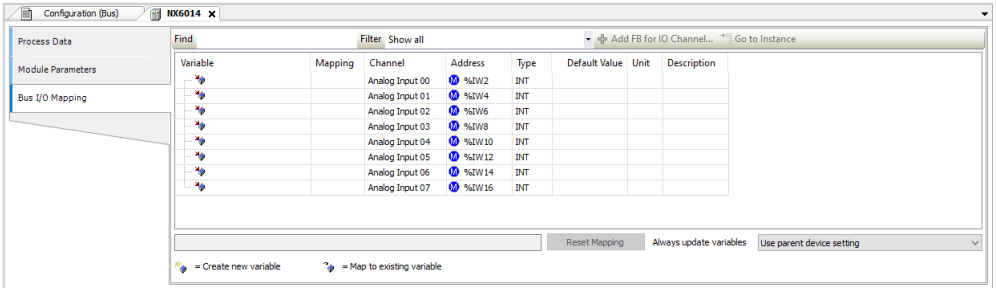

Figure 241: I/O Module Map

Note: When a mapping is done directly to a memory adress %I, %Q or %M, the size of this mapping is also defined through a modifier, for example, %IW represents 16 bits in memory area I. However, in addition to indicating the size, a data type is also created, which can be different from the type initially specified for the area. This applies, for example, when we want a 16-bit memory to have a sign representation and therefore the mapping is created with the INT type (16-bit signed). When monitoring the mapping content directly on the I/O Mapping screen, the value will be displayed according to the specified type. However, if the same mapping address is directly entered in the application, without referring to the mapping (using the variable name), the default format is the type modifier, which in all cases is unsigned.

ATTENTION

The *Always Update Variables* selection list defines the update of I/O variables.

- Use *Father* Device Configuration: Update according to *father* device configuration.
- Enable 1 (use bus cycle task if not used in another task): The MasterTool IEC XE updates the I/O variables in the bus cycle task if they are not used in any other task.
- Enable 2 (always on bus cycle task): The MasterTool IEC XE updates all variables on every cycle of the bus cycle task, regardless of whether they are used and whether they are mapped to an input or output channel.

8.4. Tag and Description

8.4.1. Modules

Each and every module inserted in the bus can receive a Tag, which will be the name of the module. In the figure below, Module Tag, available in the module properties window, the name of the module can be changed.

In the CPU, a module's tag appears on the display when the module's diagnostics button is pressed. With a second press, holding the button for more than 1 second, it is possible to view additional details about the module.

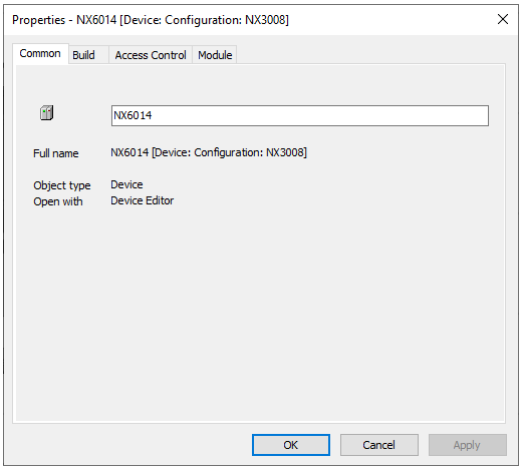

Figure 242: Module Property

8.4.1.1. I/O Points

Each I/O point can also receive a tag and description, which can help in system maintenance moment. In I/O Module Map, the *Variable* field can receive the tag of each I/O point, and there is a limit of 24 characters for this text. Also, in the same figure, in the Description field, it is possible to insert a description text for each I/O point, this text being limited to 48 characters.

8.5. Bill of Materials

Once configured the bus modules MasterTool IEC XE allows you to view a list of the modules needed to the hardware configuration as shown in the image below. This list can also be used for purchase orders.

The following modules and products are shown in this list:

- CPUs
- Input and Output Modules
- Fieldbus Interface Modules
- Power Source Modules
- Special Modules (bus expansion and redundancy)
- PROFIBUS Slaves
- PROFIBUS Input and Output Modules
- PROFIBUS Modules base (consult PROFIBUS Modules Base – Ponto Series)
- Backplanes
- Accessories
- Cables

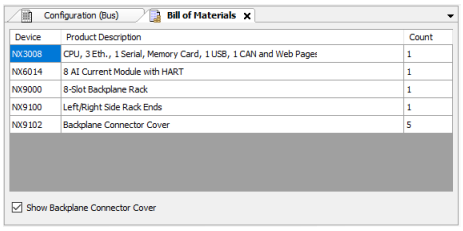

Figure 243: Bill of Materials

This list is available on the device tree for any project created from the *Standard Project*. The list shows the different types of devices that are present in the configuration with its description and quantities. This list can also be printed.

Available commands:

- *Show Backplane Connector Cover*: Add the Backplane Connector Cover device to the list of materials. The quantity is variable, according to the empty spaces in the backplane.

In projects with NX3030 CPU with redundancy, there is an additional field:

- *Show Redundant Configuration*: If the option is selected it adds accessories and devices to the list of material, which are required for redundancy.

8.6. Configuration and Consumption

Once configured the bus modules, MasterTool IEC XE allows the user to view the backplane complete configuration as shown in below.

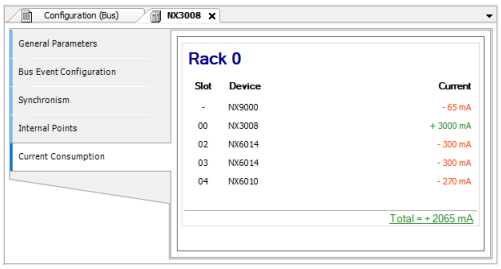

Figure 244: Configuration and Consumption

The configuration and bus consumption list is available in the device tree for any project created from the *Standard Project*. The list displays all configured devices on the bus with your type, description, and the individual identifier of each device on the bus. In addition, it provides information about the consumption of each one of them and the current balance of the project on the basis of the source model used. The information and consumption configuration can also be printed

As the rack model does not have an individual ID beyond the one displayed at the top of the screen, the *Name* field does not present a value for this device. The same goes for the visualization of power consumption in the *Edit* screen of the power supplies.

8.7. Programming Language Editors

For details see Programming Language Editors (CFC, SFC, ST, FBD/LD/IL), in IEC 61131 Programming Manual.

8.8. Declaration Editor

The textual declaration editor serves to create the declaration part of a POU object. It can be supplemented by a tabular view. Any modification made in one of the views always is immediately applied to the other one.

Depending on the current settings in the Declaration editor options either only the textual or only the tabular view will be available, or you can switch between both via buttons (Textual/Tabular) at the right border of the editor window.

Usually the declaration editor is used in combination with the programming language editors that means it will be placed in the upper part of the window, which opens when you are going to edit or view (monitor) an object in offline or online mode. The declaration header describes the POU type (for example PROGRAM, FUNCTION_BLOCK, FUNCTION ...) and might be extended by POU-global pragma attributes.

The online mode of the declaration editor is structured like that of a watch view.

Variable declaration also is done in *Global Variable Lists* and *Data Unit Types*, which however are created in separate editors.

Regard the general information on variables declaration.

8.8.1. Textual Declaration Editor

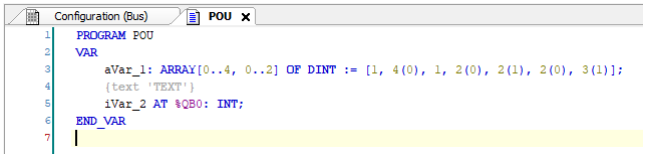

Figure 245: Textual Editor View

Behavior and appearance are determined by the respective current text editor settings in the *Options* dialogs. There you can define the default settings for highlight coloring, line numbers, tabs, indenting and many more. The usual Windows functions are available and even those of the IntelliMouse can be used if the corresponding plug-in is installed. Notice that block selection is possible by pressing <ALT> while selecting the desired text area with the mouse.

8.8.2. Tabular Declaration Editor

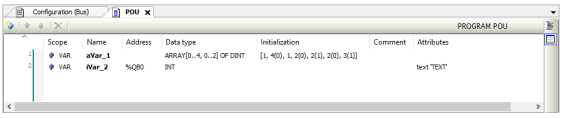

Figure 246: Tabular Editor View

The tabular view of the editor provides columns for the usual definitions required for a variables declaration: *Scope*, *Name*, *Address*, *Data type*, *Initialization*, *Comment* and *Attributes* (pragma). The particular declarations are inserted as numbered lines.

The declaration header can be edited in the *Edit Declaration Header* dialog, which opens on the same-named command.

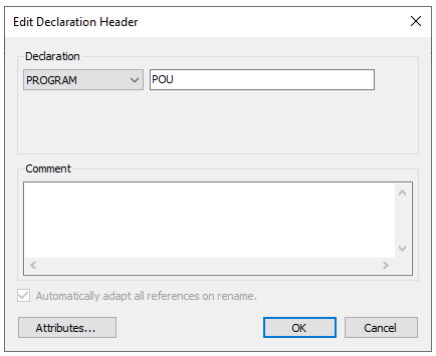

Figure 247: Edit Declaration Header

Attributes dialogs for specifying pragma instructions and attributes:

To add a new line of declaration above an existing one, first select this line and use command *Insert* from the toolbar or the context menu. To add a new declaration at the end of the table, click beyond the last existing declaration line and also use the *Insert* command.

The newly inserted declaration by default first uses scope *VAR* and the recently entered data type. Automatically the input field for the obligatory variables name will be opened where you must enter a valid identifier and close with <ENTER> or by a mouse-click on another part of the view.

Each table cell on a double-click opens the respective possibilities to enter a value.

For editing the *Scope*, the double-click will open a list from which you can choose the desired scope and scope attribute keyword (flag).

The *Data type* can be typed in directly or via the > button. The *Input Assistant* or the *Array Wizard* can be used.

The initialization value can be typed in directly or via the ... button. The Initialization *Value* dialog can be used, which is helpful for structured variables.

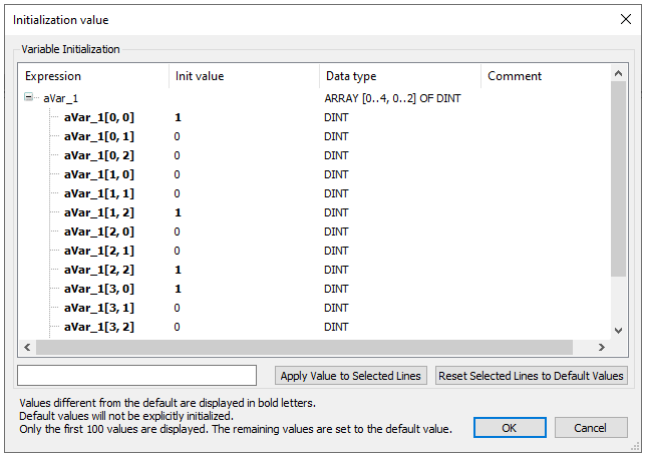

Figure 248: Initialization Value

All expressions of the variable will be displayed with the current init values. Select the desired ones and edit the initialization value in the field below the listing. Then use button *Apply value to selected lines*. To restore the default initializations, use button *Reset selected lines to default values*.

Line breaks in the *Comment* entry can be inserted via <CTRL>+<ENTER>.

The *Attributes* entries are done in the Attributes dialog where multiple attributes and pragmas can be entered in text format. They have to be inserted without enclosing braces, a separate line per each.

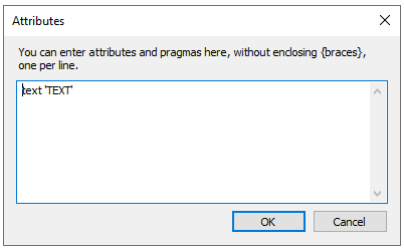

Figure 249: Attributes

Each variable is declared in a separate line, the lines are numbered.

You can change the order of lines (line numbers) by selecting a line and move it one up or down by the Move up ( ) and Move down ( ) command from the toolbar or the context menu.

The list of declarations can be sorted according to each of the columns by a mouse-click on the header of the respective column. The column which currently determines the order, is indicated by an arrow symbol: (ascending order) or (descending order). Each further click in the column header changes between ascending and descending order.

To delete one or several declarations, select the respective lines and use <DEL> or the Delete command from the context menu or from the toolbar ( ).

8.8.3. Declaration Editor in Online Mode

After login to the target system each object, which has been opened in a window already in offline mode, will automatically be displayed in online view.

The online view of the *Declaration Editor* presents a table like used in watch views. The header line shows the actual object path <*device name*>.<*application name*>.<*object name*>. It is not possible to add more than 16000 expressions in this monitoring. If this situation occur, the user will be prompted with a message. The table for each watch Expression shows the Type and current Value as well as - if currently set - a Prepared value for forcing or writing.

In case of a boolean variable the handling is even easier. You can toggle boolean preparation values by use of the <ENTER> or <SPACE> key according to the following order: If the value is TRUE, the preparation steps are FALSE > TRUE > nothing. If the value is FALSE, the preparation steps are TRUE > FALSE > nothing.

If a watch expression (variable) is an instance, for example of a function block, a plus- resp. minus-sign is preceded. Usually via a mouse-click on this sign the particular elements of the instanced object can be additionally displayed and hidden (see FB_1 and FB_2 in the figure below). Icons indicate whether the variable is an input, output or *normal* one.

As long as the default setting *Replace Constants* (*Compile options* dialog) is active, constants are indicated by an symbol preceding the value in the *Value* column.

When pointing with the cursor on a variable in the implementation part, a tooltip will show the declaration and comment of the variable. See the figure below.

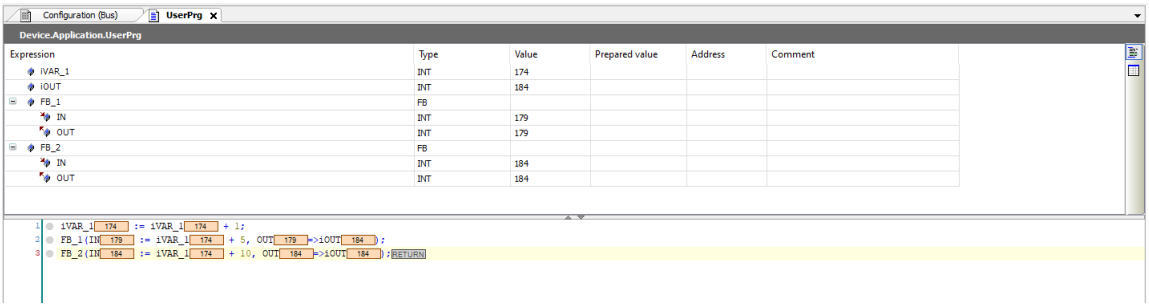

Figure 250: Declaration Editor in the Upper Part of a program object UserPrg, Online View

8.9. Device Editor

The *Device Editor* provides dialogs for the configuration of a device, which is managed in the *Devices* view window.

The editor can be opened via command *Edit Object* or by a double-click on the device object entry in the *Devices* window.

The dialog can provide tabs containing the following sub-dialogs:

- *Communication Settings*: Configuration of the connection between programming system and a programmable device (PLC).
- *Files*: Configuration of a file transfer between host and PLC.
- *Log*: Display of the PLC log file.
- *Users and Groups*: User management concerning device access during runtime (not to be mixed up with the project user management).
- *Access Rights*: Configuration of the access rights on runtime objects and files for the particular user groups.
- *Applications*: *On this tab of the generic device editor, the existing applications on the device are displayed. Depending on the system, it is possible to delete applications from the device or retrieve detailed information about them.*
- *Information*: General information on the device (name, provider, version etc.).
- *PLC Settings*: On this tab of the generic device editor, basic settings for the configuration of the PLC are made, such as handling inputs and outputs and the bus cycle task.

8.9.1. Communication Settings

This dialog is provided on a tab of the *Device* dialog, which can be opened via command Edit Object or by a double-click on the device object entry in the Devices window. It serves to configure the communication parameters for the device.

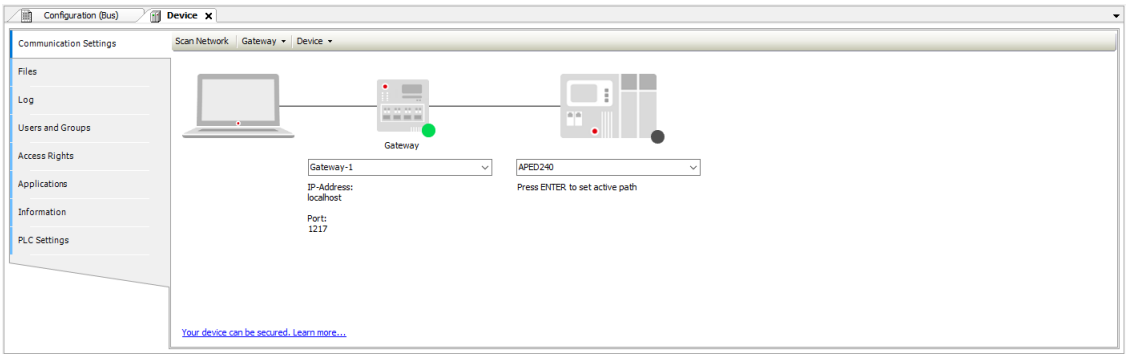

Figure 251: Communication Settings

This dialog contains three main buttons:

- *Scan Network*: This command initiates a search for available devices on your local network and updates the configuration tree accordingly. It is also triggered by double-clicking the gateway entry. The discovered devices are listed under the selected gateway entry in the tree when the command is executed.
- *Gateway*: Here, the user can configure the Gateway. The options provided to the user are:
  - *Add New Gateway...*: To add a new Gateway.
  - *Manage Gateways...*: To manage Gateways. Add, delete, move up and down.
  - *Configure the local Gateway...*: This command opens the *Gateway Configuration* dialog, allowing you to configure the block drivers for the local gateway as an alternative to manually editing the Gateway.cfg configuration file.
- *Device*: It's a series of a options to configure the device.

This tab has three connection indicators, the circles in front of the Gateway and the Device. If the circle is green, the connection is established. If the circle is red, MasterTool IEC XE cannot establish the connection. And if the circle is black, the connection status is unknown

MasterTool IEC XE Gateway Service: The CODESYS Gateway SysTray is started automatically at system start as a service and this service is represented by a control icon in the system bar. Icon indicates it's running. Icon indicates stopped Gateway. You can stop and restart the Gateway via the commands of the menu, which is accessible by a click with the right mouse-button on this icon. If necessary the service also can be started in the *Programs* menu by selecting the *Gateway* entry.

Add new gateway...: This command opens the *Gateway* dialog, where you can define a gateway to be added to the current configuration. Enter a symbolic *Name*, the *Driver* type and driver specific parameters (e.g. *IP Address* and *Port*) for that gateway. To enter the parameter values perform a double-click on the respective column field to open an edit frame. After having closed the dialog by *OK* the new gateway entry will be added in the configuration tree in the *Communication Settings* dialog.

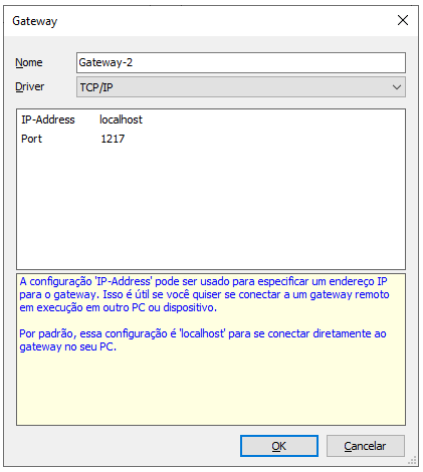

Figure 252: Add new gateway...

ATTENTION

The proper configuration of the gateway requires detailed knowledge. In case of doubt leave unchanged the default configuration settings.

Note: This page has two modes. This is the *Simple Mode*. If the user wants to change to the *Classic Mode*, it's necessary to change it in *Tools* > *Device editor* > *Communication page*.

8.9.1.1. Select Devices

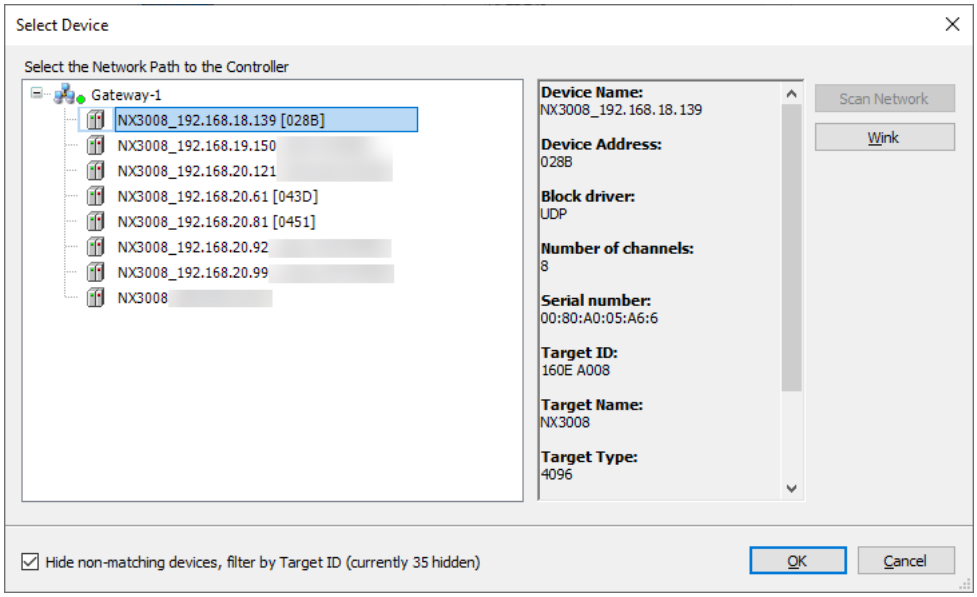

Figure 253: Select Device Dialog

In the *Select Device* dialog, after the user select a device (not double-clicking it), the user may see, in the right part of the window:

- *Device Name*
- *Device Address*
- *Block driver*
- *Encrypted Communication*
- *Number of channels*
- *Serial number*
- *Target ID*
- *Target Name*
- *Target Type*
- *Target Vendor*
- *Target Version*

Below the device tree listed in *Select Device*, there's the following commands:

- Hide non-matching devices, filter by Target ID: Checked by default. Will hide devices that aren't from the model of the device created in the project. If disabled, will show all the possible devices.

In the right part of the dialog there are the buttons:

- *Scan Network*: Enabled at a gateway is selected. Scans the networks searching for devices.
- *Wink*: Enabled when a device is selected. Some device LEDs and LCD start blinking.

8.9.1.2. Communication with Remote Gateway

The Scan Network option available in devices, only allows mapping and accessing devices that are on the same subnet of the selected gateway. This way when you add a gateway is important to know which device you want to access. If the device is present on the same subnet of the computer where the MasterTool IEC XE is installed, just add the *Gateway* and keep the localhost option in the *IP-Address* parameter. This is the case depicted in figure below where Gateway (GW0) of the computer (PC10) is being used to access the device from same subnet (PLC20). In the figure below, there are represented six different nodes in three different subnets. The nodes are addressed through IP address class C, and the communication between the different subnets is provided by the element called Router, which is also present in the representation.

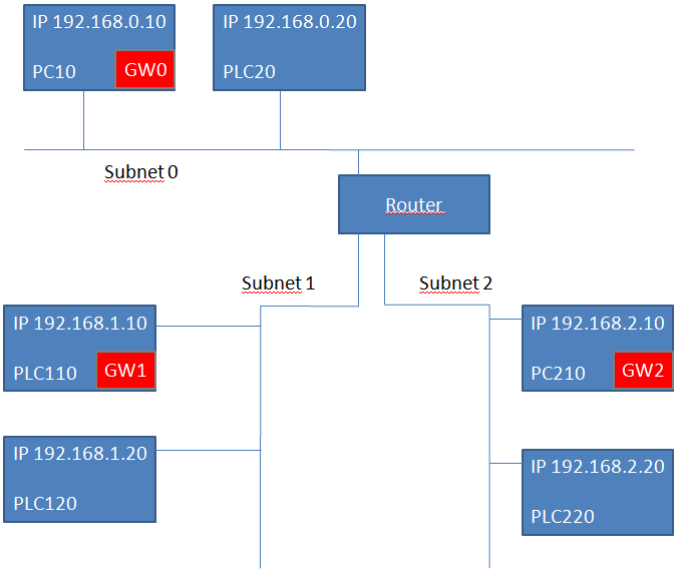

Figure 254: Remote Gateway

However, there are cases in which you need to access other subnets. For that there must always be a gateway inside the other subnet where you want to access. There are two alternatives for this situation. The first of these would add a gateway (GW2) in the computer (PC210) to access the devices that are on the same subnet (PLC220).

On the other hand, it is also possible to access a device (PLC120), in the case of a PLC Series Nexto, through another device on the same subnet (PLC110). In this case the used gateway (GW1) is part of another device known on the same subnet. This is not the ideal situation because when you access a device traffic is being generated within another one. In this way, is recommended that by accessing a remote network where there is no computer running a gateway you must configure the IP address of the target device in the device Communication Settings. In the figure above, there is no gateway running on the subnet 1. This way to access the PLC110 it is possible to configure the gateway of the device itself (GW1) which will be responsible for handling the packages with the computer on the subnet 0 for example.

In both cases, it is necessary to know what the device IP address is and put it into the parameter IP-Address when it added a new gateway. By using one of these two paths it is possible that the PC10 access PLC120 and PLC220 in different subnets than yours. For details on adding gateway see Communication Settings.

Note: In some cases, when different subnet than the PLC, you must perform a connectivity test using the command *ping*, through the Windows prompt, between the computer and the PLC to this one appears in the *Communication Settings* window.

ATTENTION

All Nexto series CPUs can be used as a remote gateway in different subnets of the computer where the MasterTool IEC XE is running. However, this operation causes performance loss of your Ethernet interfaces due to treatment and redirection of packages sent to other devices on the same network. Whenever a device is accessed on a network other than the computer where there is a gateway to redirect packets transmitted by MasterTool IEC XE , it is always recommended to use the IP address of the destination avoiding this unnecessary traffic to other equipment not involved in communication.

8.9.2. Files

This dialog is provided on a tab of the *Device* dialog (Device Editor), which can be opened for the device currently selected in the devices tree via command *Edit Object* or by a double-click on the device entry in the Devices window.

The dialog serves to transfer files between the host and the controller. This means you can choose any file from a directory of the local network to get it copied to the files directory of the currently connected runtime system, or the other way round. There is also the possibility to transfer files between the controller and a memory card, which should be inserted on the selected CPU.

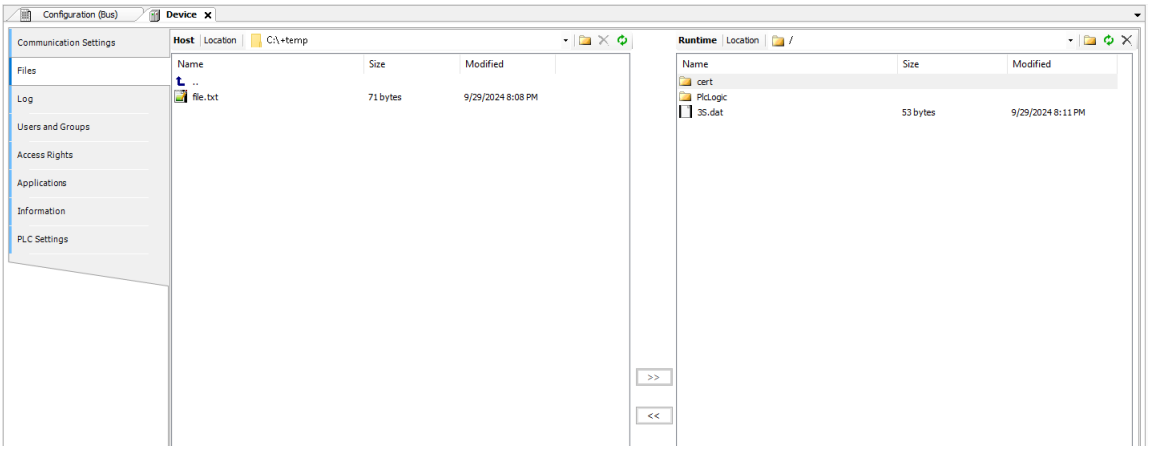

Figure 255: Files

In the left part of the dialog the files on the Host are displayed, in the right part that on the Runtime system.

In order to update the Runtime file list, use button

If you want to create a new folder where the file should be copied, use button

If you want to delete the currently selected file, use button

In the Location selection fields of Host and Runtime each enter the appropriate directories between which one or several files should be transferred. This is to be done via the selection list at the entry field or via browsing in the file system tree.

The files to be copied have to be selected in the file system tree. Multiple selection is possible, also a folder can be selected in order to get copied all contained files.

For transferring the selected files to the defined host or runtime system directory, use button » and «. To *transfer* in this context means to *copy*. So, if a file is not yet available in the *target* directory, it will be created also there. If however already a file with the given name is available and is not write-protected, this will be overwritten. In case of write-protection an appropriate message will be generated.

8.9.3. Log

This dialog is provided on a tab of the *Device* dialog (Device Editor) and serves to view the events, which have been logged on the runtime system, in the programming system. This concerns:

- Events at system start or shutdown (loaded components and their versions).
- Application download and boot project download.
- Customer specific entries.
- Log entries of I/O drivers.
- Log entries of the data server.

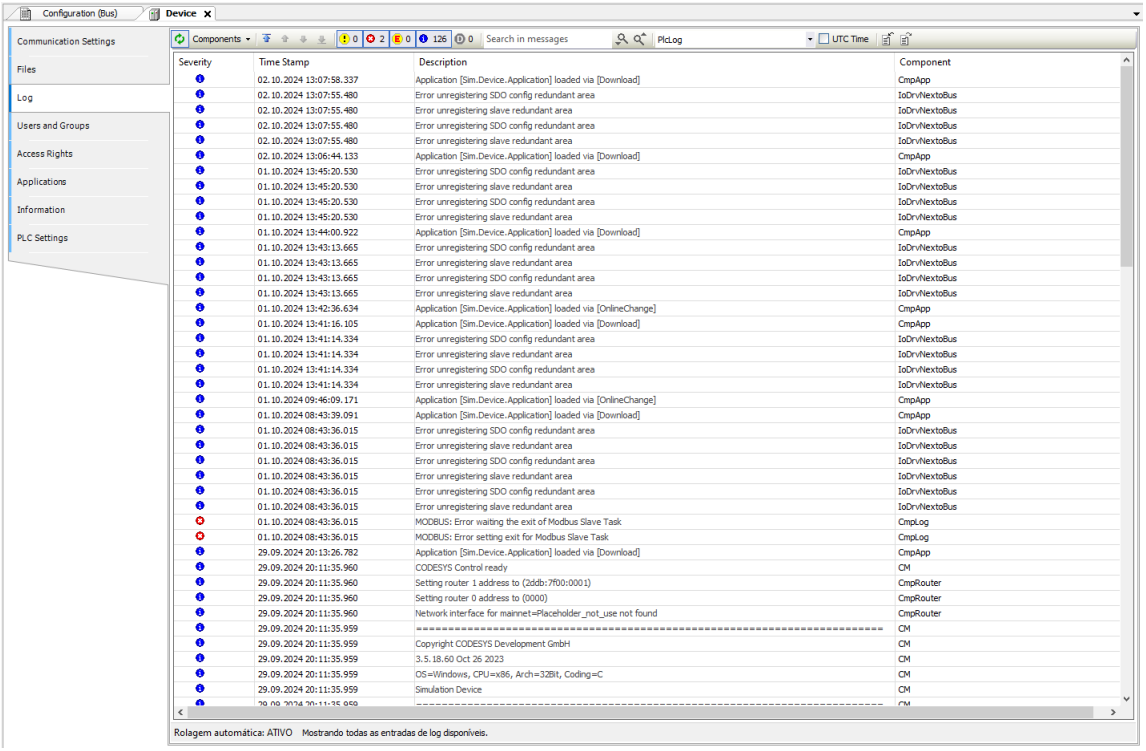

Figure 256: Log

A log entry line contains the following information:

- *Severity*: There are four categories: warnings, errors, exceptions, information. The display of the entries of each category can be switched on or off by using the corresponding button from the bar above the listing. Each button always contains the current number of loggings in the respective category.
  - : Warning.
  - : Error.
  - : Exception.
  - : Information.
  - : Debug message.
- *Time Stamp*: Date and Time: MM/DD/YY h:mm AM/PM:ss:ms.
- *Description*: Description of the event.
- *Component*: ID and name of the component.
- *Selection list of component names*: Here you can choose a particular component in order to get only displayed log entries concerning this component. The default setting is <*All components*>.
- *Logger*: The selection list provides the available log entries. The default setting is "<Default Logger>", which is defined by the runtime system and currently is identical with *PlcLog* for runtime system in Nexto Series CPUs. Currently the user can manually update the list by using button. By clicking once, it will be pressed ( ) and will update the log automatically. By clicking again, the display is disabled (new log entries will only be shown when clicking again).

Note: While the update button is pressed, the programmer and the CPU will be connected, even if the *Logout* command is executed. In this case, to provide the communication and/or sending project to other CPUs, the refresh button should be disabled.

The list can be exported to or imported from a XML-file. To export use button to get the standard dialog for saving a file. The file filter is set to *xml-files (*.xml)*. The log-file will be stored with the specified file name with extension *.xml* in

the chosen directory. To view log entries stored in an xml-file which might have been exported in this way, use button . The standard dialog for browsing for a file will open. In addition, here the filter is set to *xml-files (*.xml)*. Choose the desired log-file and its entries will be displayed in a separate window.

For clearing the current log table, that is removing all displayed entries, use button.

If option *Offline-Logging* is activated, also certain offline actions will be logged, which do not refer to the connection with the PLC.

8.9.4. Users and Groups

In this section of the generic device editor, you manage the user accounts and groups for the PLC. Based on the device's capabilities, you can configure user accounts and groups, which, in conjunction with the settings on the Access Rights tab, allow you to control access to control objects and files during runtime. To perform these actions, the controller must support user management and allow it to be edited, and you must have the necessary login credentials to access the controller.

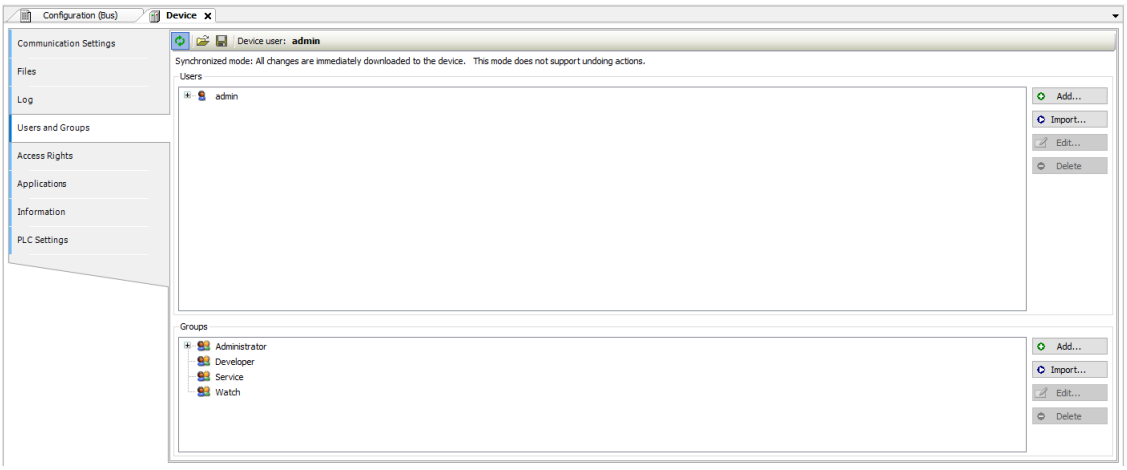

Figure 257: Users and Groups

The *Users and Groups* can be separated in three main parts, they are: toolbar, *User* block and *Groups* block.

The toolbar of tab has the following elements:

- : The synchronization between the editor and the user management on the device can be toggled on or off. When the button is not pressed, the editor remains blank or contains a configuration loaded from the hard disk. If synchronization is enabled while the editor holds a user configuration that hasn't yet been synced with the device, a prompt will appear asking how to proceed. You have two options: either upload the configuration from the device, which will overwrite the current editor content, or download the editor content to the device, replacing the existing user management configuration there.
- : When you click the button on the *Users and Groups* tab to import a Device user management file (*.dum2), a dialog box opens, allowing you to select a file from the hard drive. After choosing the file, the *Enter Password* dialog appears, prompting you to enter the password that was assigned when the file was exported. Once the correct password is provided, the user management is enabled.
  It's important to note that before version 3.34, the older Device user management file type (*.dum) was used, which did not require encryption.
- : When you click the button on the *Users and Groups* tab, the *Enter Password* dialog first appears, prompting you to assign a password to the device user management file. This password will need to be re-entered later when the file is imported to enable user management on the controller.
  After closing the password assignment dialog, a file selection window opens, allowing you to choose and import a user management configuration from the hard drive. The file type in this case is set to Device user management files (*.dum2).
- *Device user*: Username of the currently logged-in user on the device.

The following buttons are available for setting up user accounts:

Add: The dialog *Add User* opens where you can define a user name and a password. The password must be repeated in the *Confirm password* field.

ATTENTION

By opening this dialog, the *Password* and *Confirm Password* fields are going to be filled with fictional characters, the user must replace these characters with a valid password.

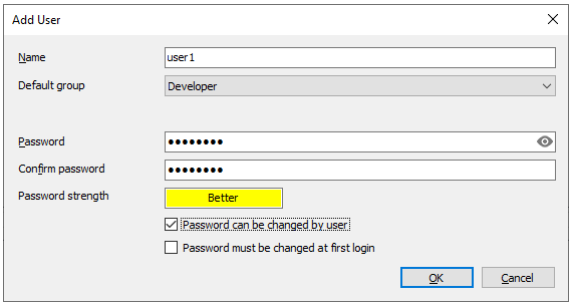

Figure 258: Add User Dialog

Import: The dialog *Import Users* opens showing all user names which are currently defined in the project user management. Select one or several entries and confirm with *OK*. The dialog *Enter* password will open where you have to enter the corresponding password as it is defined in the project user management, in order to get the user account imported to the device-specific user management.

Modify: The currently selected user account can be modified concerning user name and password. This *Edit User* <user name> dialog corresponds to the *Add User* dialog.

Delete: The currently selected user account will be deleted.

The *Group* block is composed by the following elements:

Add: The dialog *Add Group* opens where you can define a new group name and select from the currently defined users those who should be members of this group.

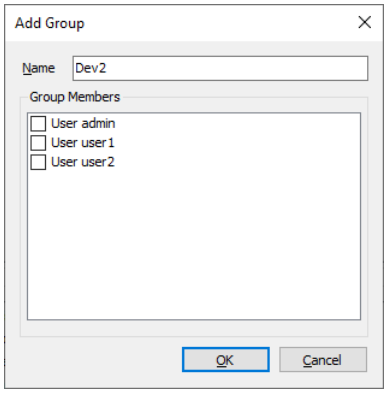

Figure 259: Add Group Dialog

Import: The dialog *Import Groups* opens where the groups currently defined in the project user management are listed. You can select one or several entries and confirm with *OK* to get them integrated in the groups list of the device-specific user management.

Modify: The currently selected group can be modified concerning group name and associated users. For this purpose the *Edit Group* <group name>, dialog opens, which corresponds to the *Add Group* dialog.

Delete: The currently selected group can be deleted.

8.9.5. Access Rights

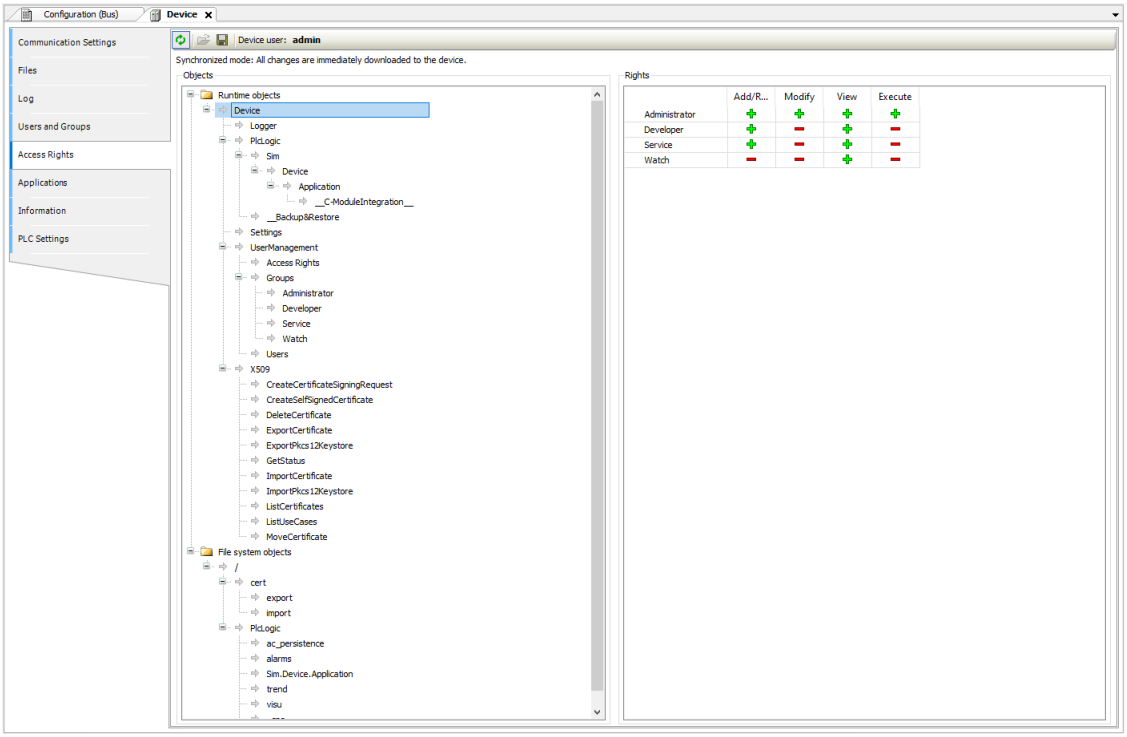

Figure 260: Access Rights

On this tab, you define the access rights of device users to objects on the controller. Similar to project user management, users must belong to at least one user group, and only these groups can be granted specific access rights.

The *Access Rights* can be separated in three main parts, they are: toolbar, *Objects* block and *Rights* block.

The toolbar of tab has the following elements:

-  : The synchronization between the editor and the user management on the device can be toggled on or off. When the button is not pressed, the editor remains blank or contains a configuration loaded from the hard disk. If synchronization is enabled while the editor holds a user configuration that hasn't yet been synced with the device, a prompt will appear asking how to proceed. You have two options: either upload the configuration from the device, which will overwrite the current editor content, or download the editor content to the device, replacing the existing user management configuration there.
-  : When you click the button on the *Access Rights* tab to import a Device rights management file (*.drm), a file selection dialog opens, allowing you to choose the appropriate file from the hard drive. Once the file is selected, the existing configuration is overwritten by the imported file.
-  : When you click the button on the *Access Rights* tab, the file type is set to Device rights management files (*.drm). In this case, there is no need to assign a password to the file before saving.
- *Device user*: Username of the currently logged-in user on the device.

Objects block:

In the tree structure, objects to which actions can be executed at runtime are listed. These objects are categorized by their source and, in some cases, organized into object groups. In the Rights view, you can configure the access options for a user group to the selected object.

- *File system objects*: Under */*, permissions can be granted to folders within the current execution directory of the controller.
- *Runtime objects*: Under *Device*, these objects manage all items with online access to the controller, and therefore require permission control.

Permissions block:

In general, subobjects inherit permissions from their root object (either *Device* or /). This means that if a user group's permission is denied or explicitly granted to a parent object, it will first affect all child objects. The table reflects the permissions for the object currently selected in the tree, displaying the rights that are configured for each user group regarding the possible actions on this object.

Possible actions for an object:

- *Add/Remove*
- *Modify*
- *View*
- *Execute*

Meanings of the symbols:

- : Access right granted explicitly
- : Access right denied explicitly
- : Access right granted through inheritance
- : Access right denied through inheritance
- : The access right was neither explicitly granted nor denied, nor inherited from the parent object, resulting in access being denied.
- No symbol: Multiple selected objects have varying access rights.

8.9.6. Applications

On this tab of the generic device editor, you can view the applications that exist on the device and, depending on the system, either delete these applications or retrieve detailed information about them.

The *Applications* tab can be separated in two main blocks: toolbar and *Applications on the PLC*

The toolbar contains the following commands:

-  *Refresh List*: The PLC is scanned for applications, and the list is updated accordingly.
-  *Delete*: Delete the selected application.
-  : Delete all applications from the list.
-  *Details*: This opens the *Details* dialog, which displays information defined for the application on the Information tab of the Properties dialog.
-  *Contents*: The requirement is that the Download the application info option is activated in the Properties of the application object on the Application generation options tab. This setting triggers additional information about the contents of the application to be downloaded to the PLC. The Contents button opens a dialog displaying additional information about the differences between the latest generated code and the application code currently on the controller, with the various POUs presented in a comparison view.

8.9.7. Information

This tab displays general information sourced from the device description file, and, if applicable, an illustration.

The informations that will be displayed are:

- *Name*
- *Vendor*
- *Categories*
- *Type*
- *ID*
- *Version*
- *Order number*
- *Description*

8.9.8. PLC Settings

On this tab, you configure the basic settings for the PLC, such as the handling of inputs and outputs, as well as the bus cycle task.

- *Application for I/O handling*: Application that is responsible for the I/O handling

In *PLC Settings* block:

- *Update I/O while in stop*: If enabled, the values of the input and output channels are refreshed even when the PLC is in STOP mode. If the watchdog detects a malfunction, the outputs are set to predefined default values. Otherwise, the values of the input and output channels in STOP mode are not refreshed.
- *Behavior for outputs in stop*: Management of the output channels when the PLC enters STOP mode.
  - *Retain values*: The current values are retained.
  - *All outputs to default value*: The default values derived from the I/O mapping are assigned.
  - *Execute program*: The management of the output values is governed by a program included in the project, which runs in STOP mode. Enter the program's name in the field on the right.
- *Always update variables*: This setting globally determines whether the I/O variables are updated in the bus cycle task. It applies to the I/O variables of the slaves and modules only if *deactivated* is specified in their update settings.
  - *Deactivated (update only if used in a task)*: The I/O variables are updated only when they are utilized in a task.
  - *Enabled 1 (use bus cycle task if not used in any task)*: The I/O variables in the bus cycle task are updated only if they are not used in any other task.
  - *Enabled 2 (always in bus cycle task)*: All variables in each cycle of the bus cycle task are updated, regardless of their usage or whether they are mapped to an input or output channel.

In *Bus Cycle Options* block:

- *Bus cycle task*: The task that controls the bus cycle is specified, with the default task defined by the device description being used. By default, the bus cycle setting of the parent bus device applies (using the cycle settings of the parent bus). This means that the device tree is searched upwards to find the next valid definition of the bus cycle task.

In *Additional Settings* block:

- *Generate force variables for I/O mapping*: This setting is only available if it is supported by the device. When compiling the application, two global variables are created for each I/O channel mapped to a variable in the I/O Mapping dialog.
- *Enable diagnosis for devices*: The CAA Device Diagnosis library is integrated into the project, generating an implicit function block for each device. If a function block for the device already exists, an extended function block is created (e.g., for EtherCAT), or another function block instance is added, which contains a general implementation of the device diagnostics. Using the function block instances, you can determine the status of all devices in the application and evaluate any errors. Additionally, the library includes functions for programmatically editing the device tree.
- *Show I/O warnings as errors*: Warnings concerning the I/O configuration are displayed as errors.
- *Enable symbolic access for I/Os*: If enabled, input and output variables (VAR_INPUT and VAR_OUTPUT) are automatically created for the device's I/O channels. To achieve this, an extended function block is generated for each slave, based on its existing function block. This automatically generated function block can be directly accessed in the application code.
  This symbolic access runs in parallel to the manually configured I/O mapping. Otherwise, access to I/O channels is not supported, and a manual setup is required. This involves creating a mapping and assigning new or existing variables to each I/O channel.

8.9.9. Memory Consumption

In this tab, the user can view the memory consumed by the PLC graphically.

As soon as the tab is opened, no information appears to the user. For this to happen, the user needs to use the *Generate Code* option in the Compile tab, and only then will the screen be updated with the correct information.

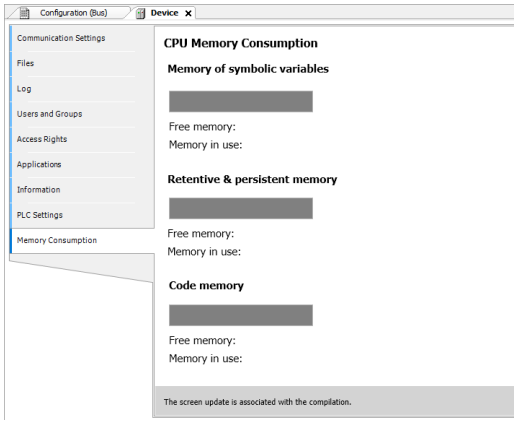

Figure 261: Memory Consumption tab with initial data

After executing the command *Generate code*:

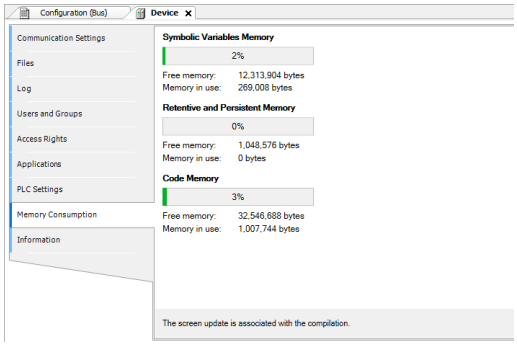

Figure 262: Memory Consumption tab after execution of the command *Generate code*

The *CPU Memory Consumption* shows three areas of memory:

- *Memory of symbolic variables*: Refers to the total space occupied by symbolic variables.
- *Retentive & persistent memory*: Refers to the total space occupied by retentive and persistent variables.
- *Code memory*: Refers to the total total program memory and the size of generated code.

The information shown in each memory area are: amount occupied (in percent and along the progress bar), *Free Memory* and *Memory in use* (both in Bytes).

As memory consumption increases, the progress bars will also grow, transitioning in color through a gradient from green to red, with yellow in between. This can be seen in the image below.

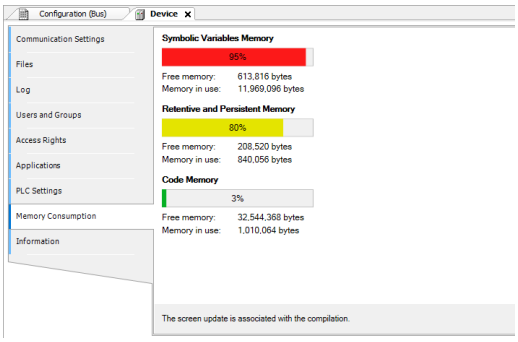

Figure 263: Gradient progress bars on the Memory Consumption tab

8.10. OPC Communication Editors

Nexto Series CPUs support the OPC DA (Open Platform Communications Data Access) communication technology. This open communication platform was developed to be the industrial communication standard. Based on a client/server architecture, it offers several advantages in project development and ease of communication with automation systems. Below are presented the configuration environments of the OPC communication in MasterTool IEC XE software. For further information on OPC Communication, consult Nexto Series CPUs User Manual – MU214605.

8.10.1. OPC DA Configuration

The PLC configuration is made within MasterTool IEC XE through an option available at *Online* menu. It is necessary that MasterTool IEC XE be executed as administrator. The figure below presents OPC Communication's configuration interface. The fields are described in the table below.

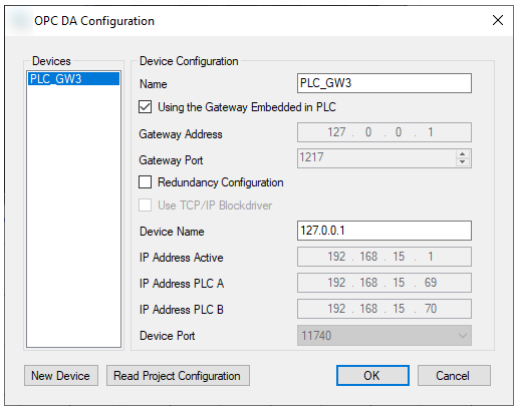

Figure 264: OPC Server Configuration

Device Configuration	Description	Factory Default	Possibilities
Name	PLC description within OPC Server configuration file. This field may have any name, but for organizational reasons, it's recommended to use the name of the project loaded in the PLC.	'PLC1'	The field is a STRING and accepts alphanumerical characters (letters and numbers) and character _. It is not allowed to start the STRING with numbers or _. It accepts 49 characters.
Gateway Address	IP address used at Gateway for communication between MasterTool IEC XE and the PLC.	127.0.0.1	1.0.0.1 to 223.255.255.254
Gateway Port	TCP Port to connect with Gateway.	1217	2 to 65534
Device Name	PLC name shown in the Device Communication Configuration tab. It is the STRING that precedes the hexadecimal value between [ ]. Only enabled when the checkbox Use *TCP/IP Blockdriver* is not marked.	'0000'	The field is a STRING and accepts any characters, just as the name of the PLC in the device Communication Configuration tab. It accepts 49 characters.
Active IP Address	PLC IP Address. Only enabled when the checkbox Use TCP/IP Blockdriver is marked. Used only when the configuration is not redundant.	127.0.0.1	1.0.0.1 to 223.255.255.254

Device Configuration	Description	Factory Default	Possibilities
PLC A IP Address	IP address of PLC A. Only enabled when the configuration is redundant. It is the primary address with which the server will communicate in case there's a failure.	127.0.0.1	1.0.0.1 to 223.255.255.254
PLC B IP Address	IP address of PLC B. Only enabled when the configuration is redundant. It is the secondary address with which the server will communicate in case there's a failure.	127.0.0.1	1.0.0.1 to 223.255.255.254
Device Port	TCP Port. Only enabled when the checkbox *Use TCP/IP Blockdriver* is marked	11740	11740 or 11739

Table 30: PLC Configuration Parameters for each OPC Server

8.10.2. Symbol Configuration Object

To configure OPC communication, just configure correctly the node and indicate the variables to be used in the communication. There are two ways to indicate which project variables are available at the OPC Server: using the *Symbol Configuration* environment or using attributes. Both cases require the object Symbol Configuration to be added; Just right-click the *Application* object and select the *Symbol Configuration*.

Below is a brief description of the fields found in this object, presented in the figure below. For further information on utilization and configuration of *Symbol Configuration*, as well as OPC communication, consult Nexto Series CPUs User Manual – MU214605, OPC DA section.

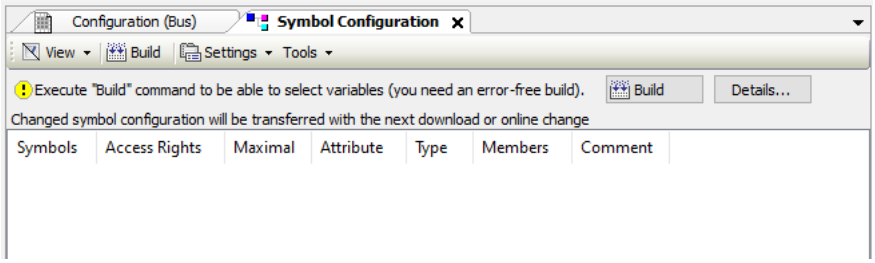

Figure 265: Symbol Configuration Object

The table below presents the description of fields in the symbol configuration screen.

Field	Description
Symbols	Variable identification that will be made available to the OPC Server.
Access Rights	Indicates the possible access level of the declared symbol. When this column is not used, it remains empty, and the access level is maximum. Otherwise, the level can be modified by clicking on the field. The possibilities are: Read only Write only Read and Write

Field	Description
Maximal	Indicates the maximal access level that can be attributed to a variable. The symbols have the same meaning as the ones in Access Rights. It's not possible to change, and it is indicated by the presence of {attribute 'symbol'}.
Attribute	Indicates whether {attribute 'symbol'} is being used when the variable is declared. When not used, this column remains empty. When used, the behavior is the following: {attribute 'symbol' := 'read'} column shows  {attribute 'symbol' := 'write'} column shows  {attribute 'symbol' := 'readwrite'} column shows  .
Type	Data type of the declared variable.
Members	A button is enabled in this column when the data type is a Struct. By clicking it, it's possible to select which structure elements are available to the OPC Server.
Comment	Comment of the variable inserted in the POU or GVL where it's declared. To appear as a comment, it must be placed one line above the declaration when in textual mode, or in the comment column when using tabular mode.

Table 31: Symbol Configuration Screen Fields Description

8.10.3. Testing OPC Communication Using Simulation

MasterTool IEC XE has a feature to connect the simulator to the OPC Server. This way, it is possible to test communication with a SCADA system without an actual PLC.

The project's configuration is just as shown before; However, PLC configuration in the OPC Server must be changed. The checkbox *Use TCP/IP Blockdriver* must always be marked. The *IP Address* must always be 127.0.0.1 and the *Device Port* must be 11739. After making these changes, just select the *Simulation* option at the project and make a *Login*.

While simulation is being executed, the simulator's variables will be available to the OPC Server. By disconnecting it, they will no longer be available. This configuration is also only possible if the gateway is configured as localhost.

Only one instance of the simulator with OPC Server can work in a computer at a time.

Limitations described at Simulation Mode still apply while using OPC Server.

8.11. Library Manager Editor

For general information on the library management in MasterTool IEC XE please see the corresponding item.

The *Library Manager* is used for including and handling libraries in a project. The installation of libraries as well as the definition of library folders (repositories) is done by the *Library Repository* dialog, which is also a component of the *Library Manager* and can be opened by the respective command in the menu bar or in the editor window.

The object is available in the devices tree to any project and is created from the *Standard Project*. It can be added to the project via command *Add Object*. It will be inserted as an object in the POUs view window or in the Devices view window assigned to a device or an application. At each of these possible positions only one *Library Manager* object can be inserted.

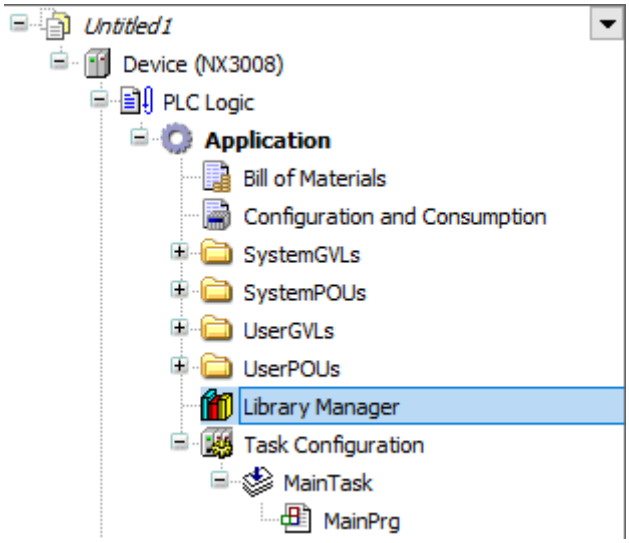

Figure 266: Library Manager Object in the POUs View Window

The Library Manager can be opened in an editor window via the *Edit Object* command or by a double-click on the object. For a description of the editor window, see below Library Manager Editor Window.

Compile messages concerning the Library Manager are displayed in a separate list in the Messages window.

8.11.1. Library Manager Editor Window

The *Library Manager* editor can be opened in an editor view window via the *Edit Object* command or by a double-click on the object.

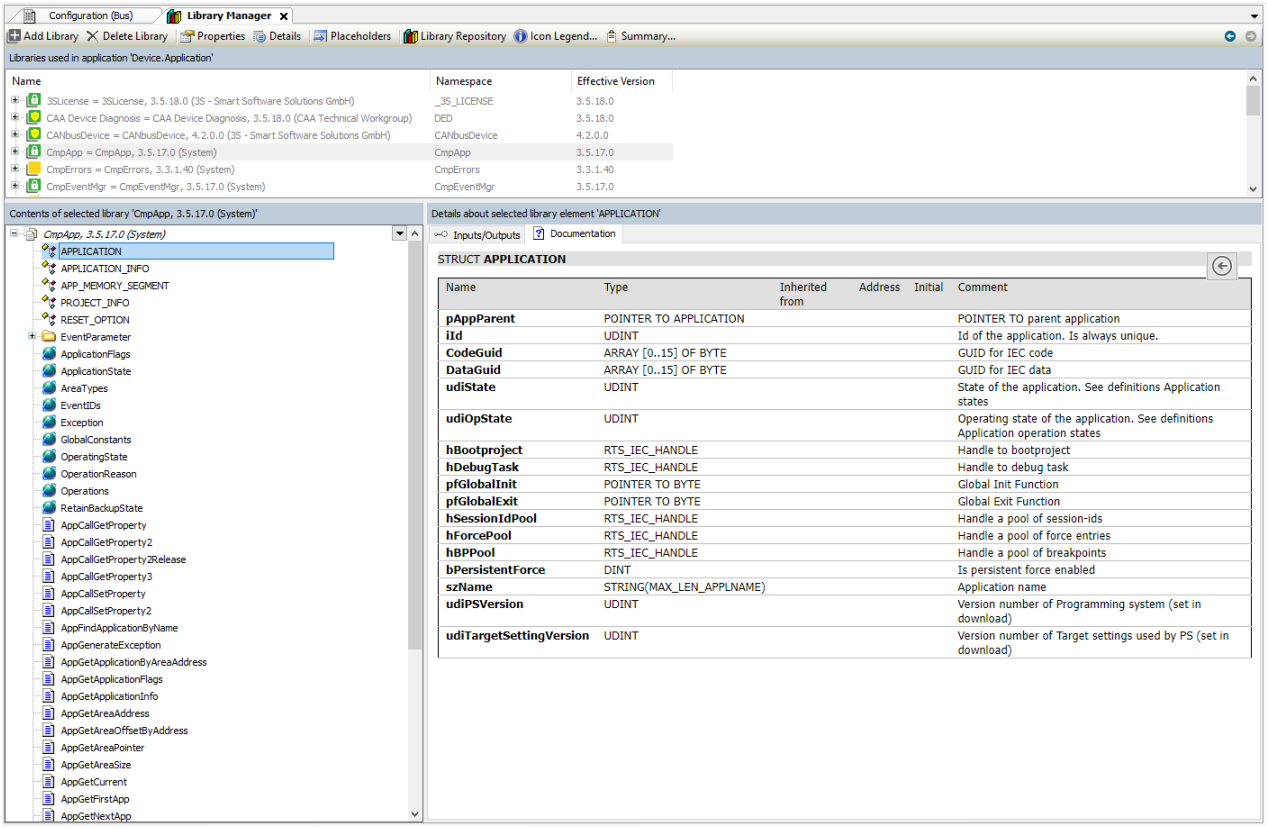

Figure 267: Library Manager Editor

8.11.1.1. Structure of the Editor Window

The upper part of the dialog displays the libraries currently included in the project. The following information is provided: *Name*, *Title*, *Version* and optionally *Company* name as it has been specified in the *Summary* dialog of the *Project Information* of the library during its creation.

- *Namespace*: The default setting for the namespace of a library is <*libraryname*> except the library explicitly has got another namespace in its *Project Information*. The library namespace must be used as a prefix of the identifier in order to uniquely access a module, which is available in multiple instances in the project. See also Library Management (IEC 61131 Programming Manual) for information on library namespaces.
- *Effective Version*: *Effective version* is that library version which will be currently used due to the definition in the library properties.

Libraries, which have been included in a project automatically by a plug-in, are, displayed grey-colored, those that have been added manually (*Add Library*) are displayed in black color.

An icon before the library name indicates the type of the library:

- MasterTool IEC XE library: Contains version information.
- The referenced file could not be found or is not a valid library: See command Try to Reload Library on IEC 61131 Programming Manual.

If a library has dependencies on other libraries (referenced libraries), those automatically will also get included - if available - and will be displayed with icon in a subtree of the entry. A subtree can be opened or closed by the respective plus- or minus-sign.

In the lower left part of the editor the modules of the currently selected library are displayed. The usual buttons for sorting, finding etc. are available for handling the modules tree.

In the lower right part the following tabs are available:

- Inputs/Outputs Tab: on the right side the components of the currently selected library module are listed in a table with variable Name, (data) Type, Address, Initial value and Comment, as defined in the library.
- Graphical: the IEC implementation of the module is viewed graphically.
- Documentation: The components of the module currently selected in the left part get displayed in a table, showing the variable Name, Data type and the Comment, which might have been added to the component declaration when creating the library. Thus inserting such comments is an easy way to automatically provide the user with module documentation.

8.11.1.2. Specific Items of the Editor Window

The following commands, which are provided in the window when one or several libraries are selected, correspond to those of the Libraries menu, which by default is available in the menu bar as long as the *Library Manager* editor is active:

- *Add library*: For including a library into the current project. Precondition: the library must be installed on the system.
- *Delete library*: The currently selected library in the libraries list will be removed from the project.
- *Properties*: Settings for the namespace and, noting that the library will later be available as a *referenced library* in another project, settings for version handling, visibility, and access.
- *Details*: Opens a dialog showing details of the currently selected library.
- *Placeholders*: Opens the *Placeholders* dialog, displaying the current resolution for editing.
- *Library Repository*: Defines library locations and allows for installing or uninstalling libraries.
- *Icon Legend...*: Opens the *Information* dialog with a legend of icons showing the current status of libraries in the integrated libraries list.
- *Summary...*: Opens the *Library Summary* dialog, displaying all libraries referenced in the project in a tree structure, along with the libraries that reference them. You can view all occurrences in the library hierarchy, and the dialog can be closed. In the Library Manager's editor, libraries in the open tree structure that reference or use the selected library are highlighted. This command is also executed by double-clicking a library. In the display of libraries, you will see the managed library's name and version, as well as the number of occurrences, indicating how many locations reference this library. By clicking the + next to a library, you can see the libraries that reference it displayed at the next level down.

8.11.2. Libraries Menu

When the *Library Manager* is active, the menu bar by default provides the Libraries menu, containing the commands for the library manager dialog.

8.12. Task Editor

Task Editor is used to handle and configure taks in MasterTool IEC XE .

8.12.1. Task Configuration

The *Task Configuration* defines one or several tasks for controlling the processing of an application program. Thus, it is an essential resource object for an application and must be available in the Devices window.

It is an obligate resource object for each application. It is available in the devices tree to any project created from *Standard Project* and can be added via the *Add Object* command

At the topmost position of a task configuration tree there is the entry *Task Configuration* (). Below there are the currently defined tasks, each represented by the task name. The POU calls of the particular tasks are not displayed in the task configuration tree.

The task tree can be edited (tasks can be added, copied, pasted or removed) by the appropriate commands usable for the devices tree. For example, for adding a new task, use command *Add Object*.

The particular tasks can be configured in the *Task Editor* dialogs, which additionally provide a monitoring view in online mode. The options available for a task configuration are target-specific.

Figure 268: Task Configuration in Devices Tree Below an Application

A Task is a time unit in the processing of an IEC program. It is defined by a name, a priority and by a type determining which condition will trigger the start of the task. This condition can be defined by a time (cyclic, freewheeling) or by an internal or external event, which will trigger the task; for example the rising edge of a global project variable or an interrupt event of the controller.

For each task you can specify a series of program POUs that will be started by the task. If the task is executed in the present cycle, these programs will be processed for the length of one cycle.

The combination of priority and condition will determine in which chronological order the tasks will be executed.

For each task you can configure a watchdog (time control); the possible settings depend on the target system.

Additionally there is the possibility to link System events (for example *Start*, *Stop*, *Reset*) directly with the execution of a project POU.

In online mode the task processing can be monitored.

8.12.2. Task Editor, Usage

By a double-click on a *Task Configuration* object in the Devices view window and by opening this object via command *Edit object* the editor window will open.

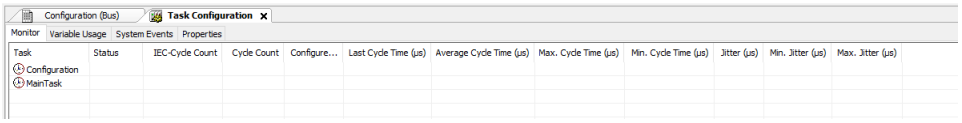

Figure 269: Task Configuration

The task configuration and a particular task can be configured in the following dialogs of the *Task Editor*, which will be opened in a tabbed window on a double-click on the respective entry in the devices tree:

- *Properties*: Dialog for information on basic task configuration settings.
- *Configuration*: Dialog for doing the configuration of a particular task, optionally extended by a *Parameter* dialog (for specific task parameters).
- *System Events*: System events dialog for linking POU calls to system events.

- *Variable Usage*: The *Variable Usage* tab provides an overview of all variables and their usage, showing in which tasks the variables are accessed.
- *Monitor*: The tab shows the status of MasterTool IEC XE tasks in online mode, along with current measurements for cycles and cycle times. The values are refreshed at the same interval as the monitoring of values from the controller.

It depends on the currently used device (target) which options are available in the configuration dialogs.

Note: Please do not use the same string function (see standard.library at IEC 61131 Programming Manual) in several tasks, because this may cause program faults by overwriting.

8.12.2.1. Properties Dialog

When the top entry in the *Task Configuration* tree is selected, the *Properties* dialog will be opened in the task editor window. The figure below shows the *Properties* window of a NX3008 CPU.

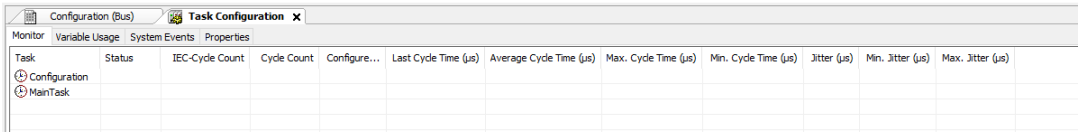

Figure 270: Task Configuration, Monitor

Information on the current task configuration as provided by the target will be displayed, for example the maximum allowed numbers of tasks per task type.

8.12.2.2. Configuration Dialog

When inserting a task (*Add Object* command) in the *Task Configuration* in the Devices view, the *Task Editor* dialog for setting the task properties will be opened.

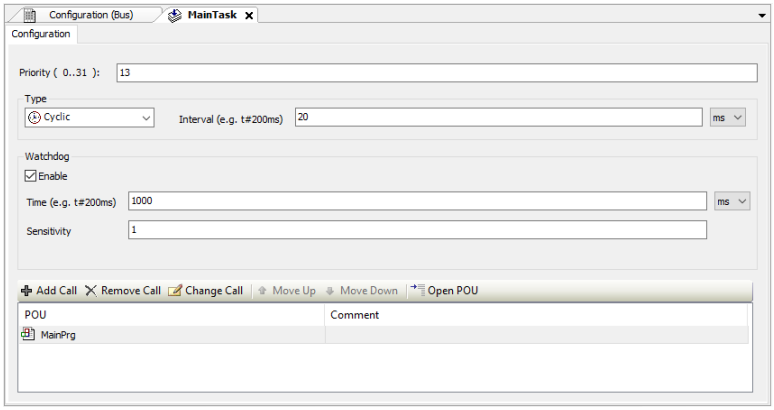

Figure 271: Main Task

Note: The task name can be modified by editing the respective entry in the devices tree. Depending on the chosen *Project Profile*, the editing of the task name may generate error in the Project building.

The dialog has the following fields:

- *Priority (0-31)*: a number between 0 and 31; 0 is the highest priority, 31 is the lowest.
- *Type*: The selection list offers the following task types.
  - *Cyclic*: The task will be processed cyclic according to the time definition given in the field Interval.
  - *Freewheeling*: The task will be processed as soon as the program is started and at the end of one run will be automatically restarted in a continuous loop. There is no cycle time defined.
  - Status: The task will be started if the variable defined in the *Event* field is true.

- *Event*: The task will be started as soon as the variable defined in the *Event* field gets a rising edge.
- *External*: The task will be started as soon as the system event, which is defined in the Event field, occurs. It depends on the target, which events will be supported and offered in the selection list (not to be mixed up with system events).

Difference between *Status* and *Event*: The specified event being TRUE fulfills the start condition of a status driven task, whereas an event driven task requires the change of the event from FALSE to TRUE. If the sampling rate of the task scheduler is too low, rising edges of the event may be left undetected.

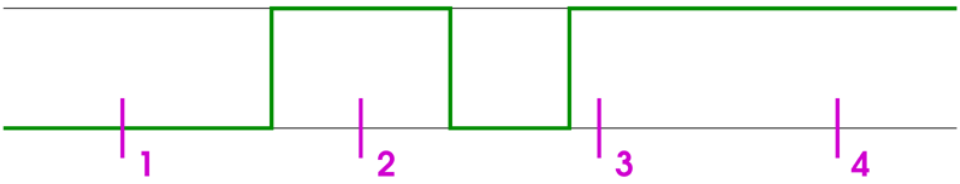

Figure 272: Resulting Behavior of the Task in Reaction to an Event (Green Line)

At sampling points 1-4 (magenta) tasks of different types show different reaction (figure above):

Behavior at point	1	2	3	4
Status	No Start	**Start**	**Start**	**Start**
Event	No Start	**Start**	**No Start**	**No Start**

Table 32: Tasks Behavior

Some entries are obligatory depending on the task choice. The *Interval* is obligatory for types *Cyclic* and *External* if the event requires a time entry (the period of time, after which the task should be restarted). If you enter a number, then you can choose the desired unit in the selection box behind the edit field: milliseconds [ms] or microseconds [µs]. Inputs in [ms]-format will be shown in the TIME format (for example *t#200ms*). Inputs in [ms] will be displayed as a pure number (for example *300*).

The *Event* is obligatory for Types *Event* or *External* and a global variable, which will trigger the start of the task as soon as a rising edge is detected. Use button ... or the *Input Assistant* <F2> to get a list of all available global event variables.

Note: If the event that is driving a task stems from an entry, there must be at least one task, which is not driven by events. Otherwise the I/Os will never get updated and the task will never get started.

- **Watchdog**: For each task a time control (watchdog) can be configured. If the target system supports an *extended* watchdog configuration, then upper and lower limits as well as a default for the watchdog time and a time definition in percent might be predefined by the device description.
  - *Enable*: When this option is activated (☑) the watchdog is enabled. This means that the task will be terminated in error status (*exception*), if the currently set watchdog Time gets exceeded, whereby there is taken account of the currently set Sensitivity. If option *Update IO while in stop* is enabled in the PLC Settings dialog, the outputs will be set to the predefined default values.

The following cases are possible:

Contiguous time overruns; the following is true:

Sensitivity	Exception in cycle...
0, 1, 2	1
3	2
...	...
N	n-1

Table 33: Sensitivity Relationship x Exception Cycle

Single time overrun: Exception if the cycle time for the current cycle is greater than (Time * Sensitivity). Example: Time=t#10ms, Sensitivity=5 > Exception: as soon as the task (once) runs longer than 50ms. This serves to detect endless loops in the first cycle.

- *Time (e.g. t#200ms)*: Watchdog time. For a description see Enable.
- *Sensitivity*: Number. For a description see Enable. If the task is MainTask the sensitivity is set to 1.

Note that a watchdog may be disabled for particular by use of the functions provided by the library CmpIecTask.library; this may be useful for cycles requiring more time as usual due to initialization processes.

After declaring an appropriate variable for the handle of the task (of type RTS_IEC_HANDLE),

```
hIecTask: RTS_IEC_HANDLE;
```

The disabling (and succeeding reenabling) can be handled by employing the interface functions in the following manner:

```
hIecTask := IecTaskGetCurrent(0);
IecTaskDisableWatchdog(hIecTask);
// Watchdog-protected code
IecTaskEnableWatchdog(hIecTask);
```

- The POUs, which are currently controlled by the task, are listed here in a table with POU name and optionally a comment. Left to the table there are commands for editing:
  - In order to define a new POU open the *Input Assistant* dialog via command *Add*. POU. There choose one of the programs available in the project.
  - In order to delete a call, select it in the table and use command *Remove* POU.
  - Command *Open* POU opens the currently selected program in the corresponding editor.
  - In order to replace a program call by another one, select the entry in the table, open the *Input Assistant* and choose another program.

The sequence of the listed POU calls from up to down determines the sequence of execution in online mode. Via the commands *Move up* and *Move down* the currently selected entry can be shifted within the list.

8.12.3. Task Editor in Online Mode

Rules of *Task Editor* and *Task Configuration* in online mode.

8.12.3.1. Which task is being Processed?

For the execution of the tasks defined in the Task Configuration the following rules apply:

- That task is executed, whose condition has been met; that is, if it is specified time has expired, or after its condition (event) variable exhibits a rising edge.
- If several tasks have a valid requirement, then the task with the highest priority will be executed.
- If several tasks have valid conditions and equivalent priorities, then the task that has had the longest waiting time will be executed first.
- The processing of the program calls will be done according to their order (top down) in the task editor. If a program is called which with the same name is available assigned to the application in the device tree as well as in a library or project-globally in the POUs window, that one directly assigned to the application will be executed.

8.12.3.2. Monitor, Online View of Task Editor

When the top node in the *Task Configuration* tree is selected, besides the *Properties* dialog on a further tab the *Monitor* dialog is available.

In online mode it shows the status and some current statistics on the cycles and cycle times are displayed in a table view. The update interval for the values is the same as used for the monitoring of PLC values.

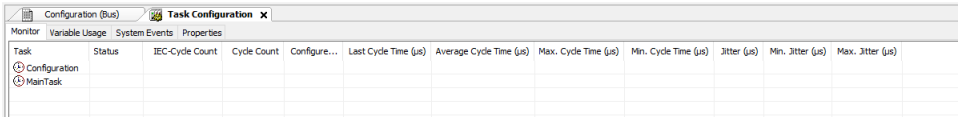

Figure 273: Task Configuration

Note: In the Simulation mode, there may be limitations and different behaviors. For further information see Simulation Mode.

For each task the following information is displayed in a line.

Information	Description
Task	Task name as defined in the task configuration.
Status	▪ Possible entries: • Not created: has not been started since last update; especially used for event tasks. • Created: task is known in the runtime system, but is not yet set up for operation. • Valid: task is in normal operation. • Exception: task has got an exception.
IEC-Cycle Count	Number of run cycles since application start. *0* if the function is unsupported by the system.
Cycle Count	Number of completed cycles. May equal IEC-Cycle Count if the system counts cycles when the application is idle.
Last Cycle Time (µs)	Last measured runtime in µs.
Average Cycle Time (µs)	Average runtime of all cycles in µs.
Max. Cycle Time (µs)	Maximum measured runtime of all cycles in µs.
Min. Cycle Time (us)	Minimum measured runtime of all cycles in us.
Jitter (µs)	Last measured jitter* in µs.
Min. Jitter (µs)	Minimum measured jitter* in µs.
Max. Jitter (µs)	Maximum measured jitter* in µs.

Table 34: Tasks Behavior

*** Jitter**: Time between when the task was started and when the operating system indicates that it is running.

When the cursor is placed on a task name field, the values can be reset to 0 for the respective task by the Reset command available in the context menu.

8.12.4. Task Configuration Commands

Commands to handle *Task Configuration*.

8.12.4.1. Add Task

This command adds a new task. If currently the root entry is selected, the new task will be appended at the end (bottom) of the current task list. If currently another task entry is selected, the new task will be inserted above this entry.

By default the new task will be named *Task*, *Task_1*, *Task_2* ... *Task_<n>*. If already a *Task* exists, you can rename the task in the *Task Properties* dialog.

When inserting a task, the dialog for setting the Task properties will be opened: The maximum number of tasks is defined according to the used CPU model.

Note: Online changes cannot be applied when task settings are changed or when tasks are added or removed.

8.13. Watch List Editor

Standards, rules and best practices and configuration of *Watch View* and *Watch List Editor*.

8.13.1. Watch View / Watch List Editor

A watch view can be opened via the *Watch* command submenu (by default in the *View* menu). It provides an editor for creating watch lists.

A watch list can be used to define and monitor a list of expressions of various objects and to write or force values to those expressions on the controller in online mode. By default four individual watch lists can be set up in the watch views *Watch 1*, *Watch 2*, *Watch 3*, *Watch 4*. The *Watch All Forces* option in online mode always gets filled automatically with all currently forced values of the active application.

8.13.2. Create Watch List

To set up a Watch <n> list in the watch view perform a mouse-click in a field of the Expression column to open an edit frame. Enter the complete path for the desired watch expression.

Syntax for watch expression:

- <device name>.<application name>.<object name>.<variable name>.
  - Example: *Device.Application.UserPrg.iVar_1*.

The type of the variable will be indicated by an icon: = input, = output and = normal. If a comment has been added to the declaration of a variable, it will be displaced in column *Comment*.

After having closed the edit frame, the Type will be added automatically in column *Type*, and the *Address* will be filled in case the variable is attributed to an address.

The *Value* column will be used in online mode for displaying the current value of the expression.

To appoint a prepared value to a variable you may click in the assigned field of the column *Prepared Value* and directly type in the desired value.

In case of a boolean variable the handling is even easier. You can toggle boolean preparation values by use of the <ENTER> or <SPACE> key according to the following order: If the value is TRUE, the preparation steps are FALSE > TRUE > no entry. If the value is FALSE, the preparation steps are TRUE > FALSE > no entry.

Do the same for the desired further expressions/variables in further lines. See an example in the next picture, which shows the watch view in offline mode. Notice that in case of a structured variable, like here the function block instance, the particular instance components automatically get added when you enter the instance name (see in the example: *Device.Application.UserPrg.fbinst*). They can be displayed/hidden in a fold via a mouse action on the plus-/minus sign.

There's also the columns *Application* and *Execution Point*. The *Application* column refers to the application where the variable is set, and *Execution Point* the type of monitoring that is being made.

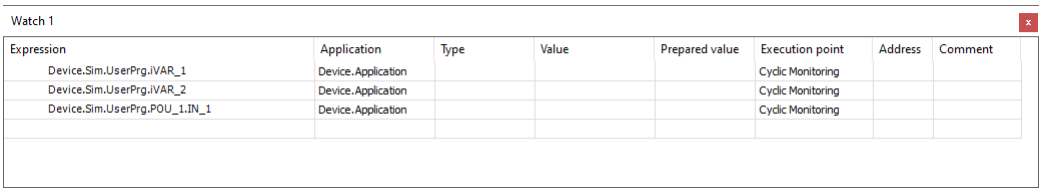

Figure 274: Watch View in Offline Mode

8.13.3. Watch List in Online Mode

Settings of *Watch List* in online mode.

8.13.3.1. Monitoring

A watch list Watch<n> in online mode shows the current value of a variable in the *Value* column.

For a description on how to set up such a watch list and how to handle folds in case of structured variables, see Create Watch List.

ATTENTION

When the monitored values represent direct mappings addresses areas of %I, %Q and %M, consistency is not performed, it only exists for the device where monitoring is being made. Since the projects can be made to any CPU models with different memory sizes, the addresses outside the range will not return valid monitoring values and, in some cases, the value read can be 0.

8.13.3.2. Write Values

In column *Prepared Value* you can enter a desired value which will be written or forced to the respective expression on the controller by command *Write Values* or *Force Values* like usable in other monitoring views (for example declaration editor).

8.13.3.3. Watch All Forces

This is a special watch list view, which in online mode gets automatically filled with all currently forced values of the active application. Each Expression, Type, Value and Prepared value will be shown, like in the online view of a Watch<n> list.

You can unforce values by one of the following commands available via the *Unforce...* button:

- Unforce all selected Expressions, without modifying the value.
- Unforce all selected Expressions and restore the variable to the value it had before forcing it.

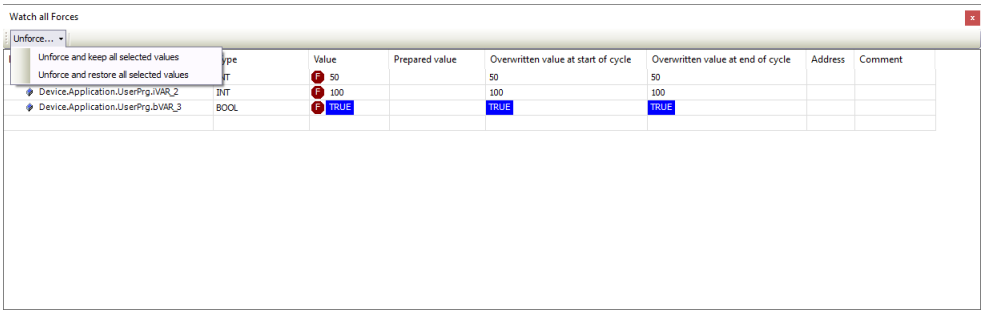

Figure 275: Watch All Forces

8.14. MODBUS Editor

When a MODBUS device is added to the project, it appears under the CPU or the selected interface, depending on the MODBUS type. Afterward, several settings must be configured to ensure correct network operation.

The configuration of these settings through the *Device Editor* are described in the current CPU manual.

8.15. PROFIBUS Editor

When we add the PROFIBUS Master NX5001 to bus, the same appears in the device tree, below the CPU and enable several settings that must be made to the correct operation of the network.

The parameters of the PROFIBUS protocol screens configuration are described in the manual MU214601.

Note: Online changes cannot be applied when network or PROFIBUS modules parameters have changed or modules are added or removed from the configuration.

8.16. CPU Editor

The CPU-related parameters are configured as the screen below. The screen is in the device tree.

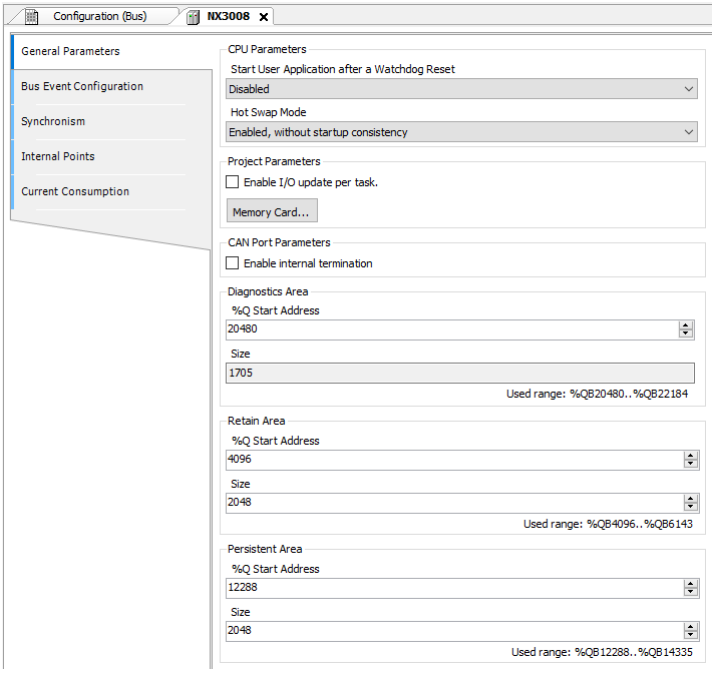

Figure 276: CPU Editor

For information on parameters, features and configuration possibilities see the current CPU manual

Note: Online changes cannot be applied when CPU parameters have changed.

8.17. Serial Interfaces

The COM1 and COM2 serial interfaces are configured in the following screen. They are located in the devices tree, below the CPU.

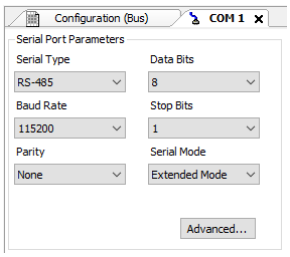

Figure 277: Serial Interfaces

For information on parameters, features and configuration possibilities see the current CPU manual.

8.18. Ethernet Interfaces

The Ethernet interfaces are configured in the following screen. They are located in the devices tree, below the CPU.

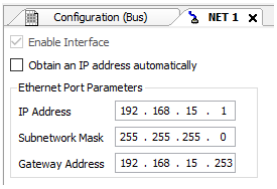

Figure 278: Ethernet Interfaces

8.19. PID Control Editor

This object allows inserting a PID controller for easy editing in an application of MasterTool IEC XE . The following will be presented all the features found in the *PID Control object*. Among them may be mentioned as examples: graphical display of the process, setting the parameters of the controller, automatic tuning procedure, setting of variables used by the controller, etc.

8.19.1. Insert PID Control Object in the Application

A *PID Control* object can be added to the application by *Add Object* command on context menu of the *Application* object, as you can see in the figure below.

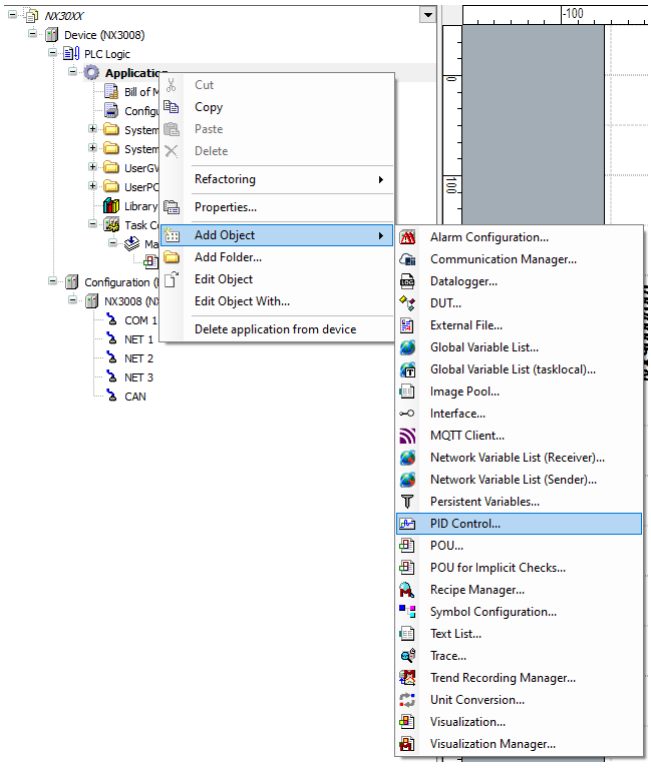

Figure 279: Steps to inserting the PID Control object in an application

When you add a *PID Control* object is inserted a Program type POU into the project. This program contains a function block of PID type as well as all logical and necessary parameters for its use. Within the object can be configured as variables that are used as inputs and outputs as well as the sampling time used in the control.

8.19.2. Graphical Environment

The graphical environment of the PID Control object is formed by a screen composed of two tabs:

- *Settings & Chart*: This is the main tab, where are configured the main parameters and where is located the trend chart.
- *Advanced Settings*: in this tab are contained minor PID loop settings.

The figure below displays the graphical environment of the *PID Control* object with its main tab open. First of all can be observed the trend chart, the bar graphs, the possibility to perform settings of some parameters of PID controller, among others. These and all other features of this object are presented in this document.

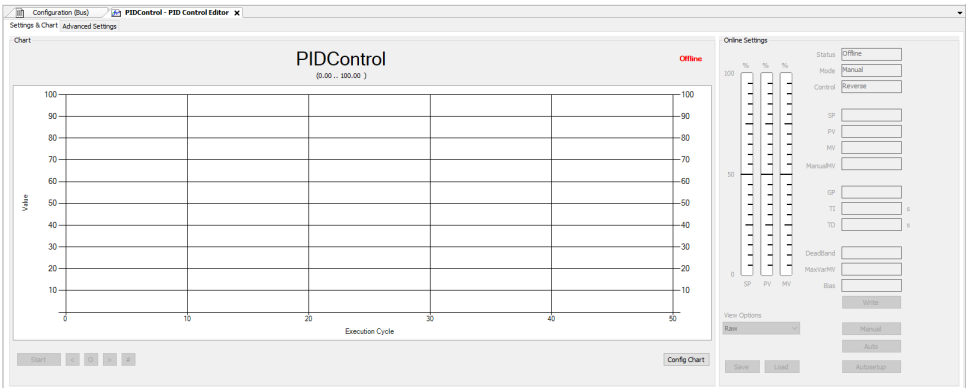

Figure 280: Graphical environment of the PID Control object

The graphical environment illustrated by the figure above can be accessed by double-clicking with the left mouse button on the *PID Control* object, located in the treeview, and then selecting the graphical environment, as shown in the below.

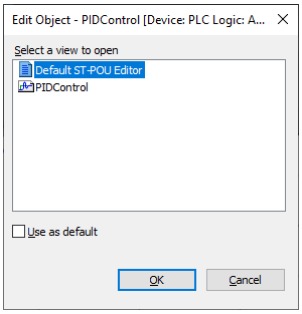

Figure 281: Access to the graphical environment of PID Control object

8.19.2.1. Tab Settings and Chart

The Graphical environment of the PID Control object presented the main work tab of the *PID Control* object. According to the illustrated in this figure, this tab is comprised of two groups: *Chart* and *Online Settings*.

8.19.2.1.1. Group: Chart

This group is responsible for displaying the process trend chart, including some operations and/or settings in this graph. The figure below presents this group.

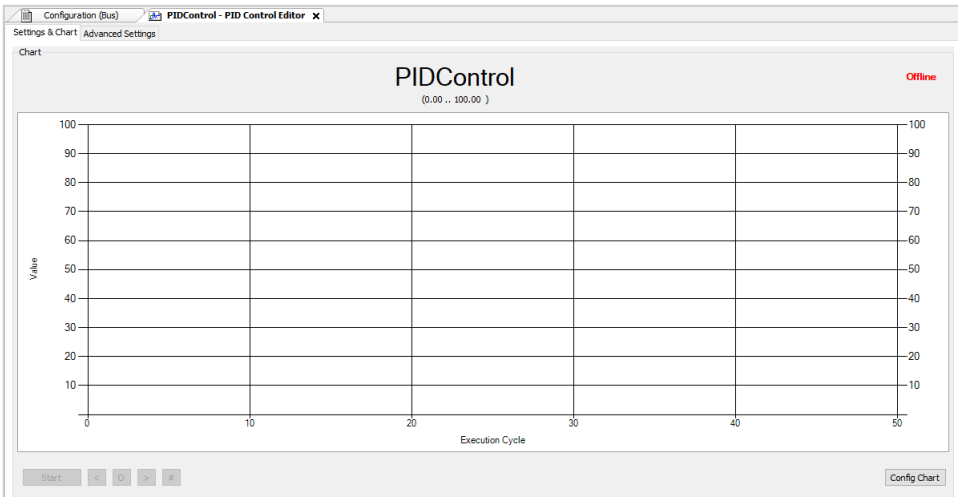

Figure 282: Group Graphic

Above the trend chart is displayed in highlight the name of PID controller in accordance with the name assigned to the *PID Control* object in the treeview window Devices. Just below is the scale of engineering of process variable (PV).

In the upper-right corner, still above the chart, the group Chart has an indicator that allows the user to recognize if the PLC is in *Offline* or *Online* mode.

The values presented in the trend chart are always displayed in percentage. This chart is composed of three pens representing the following variables:

- *SP*: controller reference value, always drawn in green.
- *PV*: controller process variable, always drawn in red.
- *MV*: controller manipulated variable, always drawn in blue.

Below the trend chart are located buttons that provide functionalities that are applied to the chart. The figure below shows these buttons.

Figure 283: Buttons of Features of the Trend Chart

The *Start* button is used to start the process monitoring. In this case, the pens of the chart will draw the dynamic behavior of variable SP, PV and MV. After the start of monitoring the name of this button changes to *Stop*, where the monitoring can be finalized. After you stop monitoring the button name returns to *Start*.

The *Start* button and all other buttons that are located immediately on your right are only enabled with the PLC in *Online* mode. These other buttons have the following functions:

- < : shift the graph to the left.
- > : shift the graph to the right.
- *O* : back to the normal position of the chart.
- # : run chart autofit operation.

By clicking with the right mouse button on the trend chart appears a context menu, as shown in the figure below:

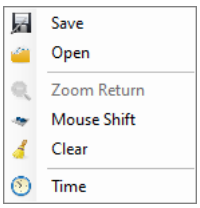

Figure 284: The Context Menu of the trend chart

- The *Save* option is enabled only when there is some information present in the trend chart. This option allows them to be stored in the file **.CSV* the data drawn by graphic pens.
- The *Open* option allows data previously saved on file **.CSV* via the option *Save* from being read and drawn again in the trend chart. This option is disabled when the process monitoring starts (initialized by the *Start* button).
- The *Zoom Return* option enables zooming in the chart. This option is enabled when there is information drawn on the graph.
- The *Mouse Shift* option allows, from the mouse, to perform shifts in trend chart. This option is enabled when there is information drawn on the graph.
- The *Clear* option allows the graph to be cleared, erasing all the information contained in it. This option is enabled when there is information drawn on the graph.
- The *Time* option allows you to switch the data type of the *x* axis between run-cycle and time in seconds.

8.19.2.1.2. Group: Chart - Chart Configuration Window

The *Chart Configuration* window is accessed by pressing the *Config Chart* button from group *Chart*, located in the *Settings & Chart* tab. This button is enabled only with the PLC is in *Offline* mode.

The *Chart Configuration* window allows you to configure some visual characteristics of the trend chart. The figure below shows this window.

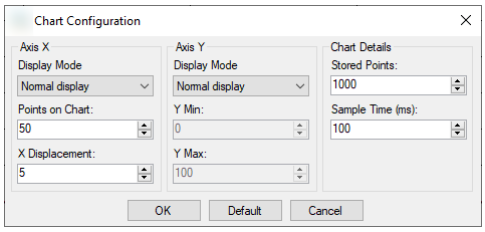

Figure 285: Chart Configuration Window

It can be observed that this window is formed by the *X Axis*, *Y Axis* and *Chart Details* groups.

In the group *X Axis* is possible to configure how the axis *x* will be displayed, with following options:

- *Normal display*: Only the last k monitoring points are displayed in the chart, where k is configured in the field *Points on Chart*.
- *AutoFit*: All monitoring points are presented in the trend chart.

The *X Displacement* field specifies the step offset of the < and > buttons, located just below the trend chart.

In the group *Y Axis* is possible to configure how the axis *y* will be displayed, with following options:

- *Normal display*: The *y* axis will be displayed in the range of 0% to 100% (fields *Y Min* and *Y Max* remain disabled).
- *AutoFit*: The *y* axis will automatically adjust to display the data collected by monitoring (fields *Y Min* and *Y Max* remain disabled).
- *Personalized*: The range of values displayed by the *y* axis can be customized using the *Y Min* and *Y Max* fields.

In the group *Chart Details* is possible to configure the details of how to the chart will be displayed and stored, with following options:

- *Stored Points*: Sets the maximum buffer storage points on the graph for each process variable, after that limit points are discarded.
- *Sample Time (ms)*: Sets the time that each point on the graph is updated for each process variable.

The changes will only be confirmed by pressing the *OK* button. If you need to return to the default setting, the Default button must be pressed.

8.19.2.1.3. Group: Online Settings

This group is responsible for showing and enable the configuration of main parameters of *PID controller*. The features of this group are only enabled with the PLC in *Online* mode. The figure below displays details of the *Online Settings* group.

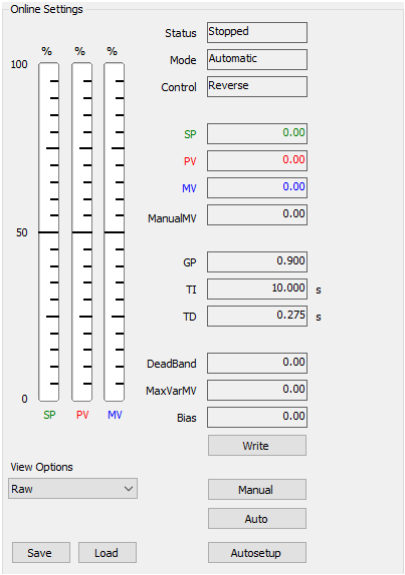

Figure 286: Online Settings

In the upper left corner of the figure above, are the bar graphs that display the current values of the variables SP, PV and MV in percentage.

On the right side of the Online Settings group are the non-editable fields:

- *Status*: Informs the PLC status, can take: *Offline*, *Stopped* or *Run*.
- *Mode*: Informs whether the PID controller is configured in *Automatic* or *Manual* mode.
- *Control*: Informs right direction of the MV's action to provide a negative feedback. When in *Direct* control, it indicates that MV should increase in response to an increase in PV, when in *Reverse* control, it indicates that MV should decrease in response to an increase in PV.

Just below, the SP field allows the user to view the current reference value of PID controller, as well as its fit when the controller is operating in automatic mode.

In the *PV* field is possible to show the value of the process variable of PID controller. This field does not allow editing.

In the *MV* field you can show the value of the variable manipulated by PID controller. This field does not allow editing.

When the PID controller is in automatic mode, the *ManualMV* field does not allow editing. However, when the controller is in manual mode, the value of the variable MV can be adjusted through this field.

The *GP*, *TI* and *TD* fields allow editing of parameters proportional gain, integrative time and derivative time of the PID controller.

In the *DeadBand*, *MaxVarMV* and *Bias* fields are set, respectively, the dead band, maximum variation allowed for the MV variable and the offset added to MV.

The Write button is responsible for sending to the PLC all parameters that have been modified, realizing the change of PID controller parameters. For further details about the write operation see Write Parameters Operation.

Manual and Automatic buttons change the mode of operation of the controller.

The *Autoconfigure* button opens the auto-tuning procedure window. For further details about the execution of auto-tuning procedure see Auto-Tuning Procedure.

The field *View Options* is used to control the view of the values of the parameters and variables. The possible values are:

- *Raw*: displays the values such that is in PLC.
- *Percent*: displays the values in percentage, in the range of 0% to 100% within the range of minimum and maximum values of the parameter or variable.
- *Engineering*: displays the values in the form of engineering scale set to the parameter or variable.

The fields affected by the view options are: *SP*, *PV*, *MV*, *ManualMV*, *DeadBand*, *MaxVarMV* and *Bias*. These fields are affected only when displayed in the text box. The trend chart and the bar graphs are always shown in percentage.

The engineering unit displayed next to the text box of the fields also changes according to the selection of the field *View Options*, showing the engineering unit (if configured) in percentage % or nothing if the option is Raw.

For the *MV*, *ManualMV*, *MaxVarMV* and *Bias* fields does not make sense to view in engineering scale. In this case, it will be displayed as a percentage.

The *Save* button saves on file **.CSV* current controller settings.

The *Load* button reads from the file **.CSV* with previously saved settings and loads them into the controller.

The *Save* and *Load* buttons allow saving and loading controller parameters, for example, after a PLC maintenance procedure. To load controller parameters that were previously stored in a **.CSV* file click the *Load* button. After the selection of the desired **.CSV* file, a small window (the image below) will be presented to the user, allowing parameters loading procedure.

The values of parameters can be saved and loaded into *PID Control* object. By pressing the *OK* button, the parameters marked will be automatically loaded into the PLC, reconfiguring the controller.

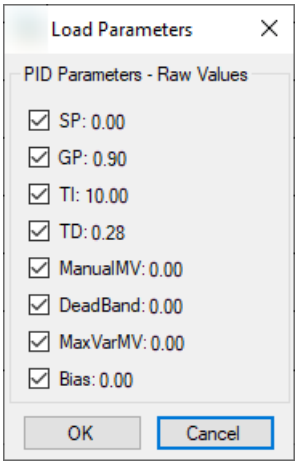

Figure 287: Parameters Selection Window

8.19.2.1.4. Write Parameters Operation

The writing or editing operation of PID controller parameters in PLC is performed from the *Write* button located in the *Online Settings* group.

The background of the fields of *Online Settings* group that allow editing, when they change their values, pass to the blue color and the font to white. This indicates that the value of the parameter or variable has been modified and the new value has not yet been sent to the PLC. By pressing the button *Write* all parameters and variables in this condition and with no error messages are sent to the PLC and the background and font colors are restored.

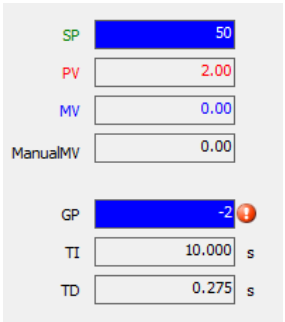

Figure 288: Parameters Editing Example

The above demonstrates that the fields *SP* and *GP* have changed, but have not yet had their new values sent to the PLC. Notice that the *GP* field displays an error message, because its value is negative (the error message may be displayed by passing

the mouse over the exclamation mark). In this case, when the *Write* button is pressed, only the values of the fields that have no errors are actually modified in PLC. The below displays the preview of these fields after the *Write* button is pressed.

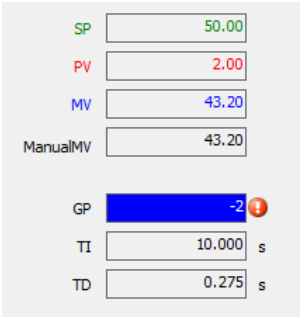

Figure 289: Viewing Parameters After Writing

If the fields value have been changed and their value has not yet been sent to the PLC, it is possible to restore its current value by clicking with the right mouse button on the field and then selecting *Actual value*. This operation is displayed in the below.

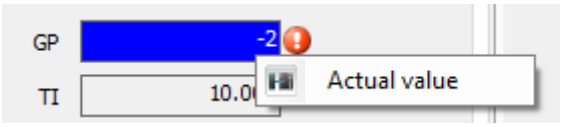

Figure 290: Restoring the Value of a Field

8.19.2.2. Tab Advanced Settings

After insertion of the *PID Control* object in the application, the first step to use the PID controller is to adjust the PID loop settings according to the application. To achieve this the tab *Advanced Settings* must be accessed. The figure below displays this tab.

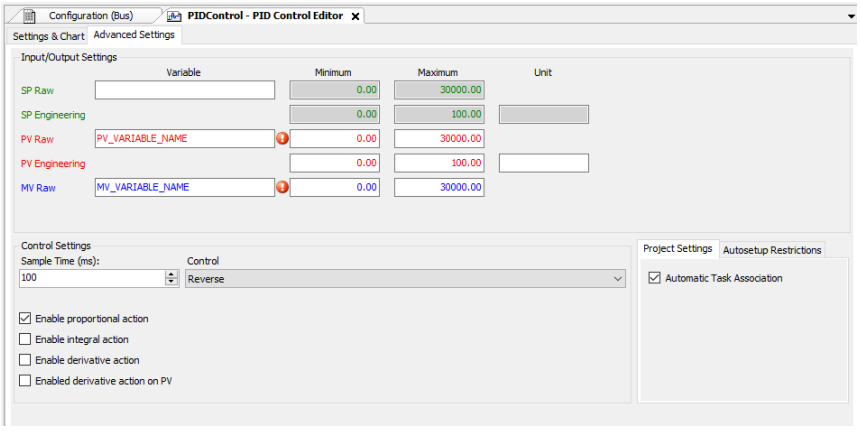

Figure 291: Advanced Settings Tab

Observing the figure above, note that the possible settings on this tab are divided into two groups: *Input/Output Settings* and *Control Settings*, and two tabs *Project Settings* and *Autosetup Restrictions*.

It is important to notice that all changes made in the Advanced Settings tab should be made with the PLC in *Offline* mode. After parameters changing it is necessary to load the project into the PLC. If the PLC is in *Online* mode all fields will be disabled, not allowing editing.

8.19.2.2.1. Group: Input/Output Settings

This group is used to configure the operation of the input and output of the PID, PV and MV, respectively. The figure below shows the group.

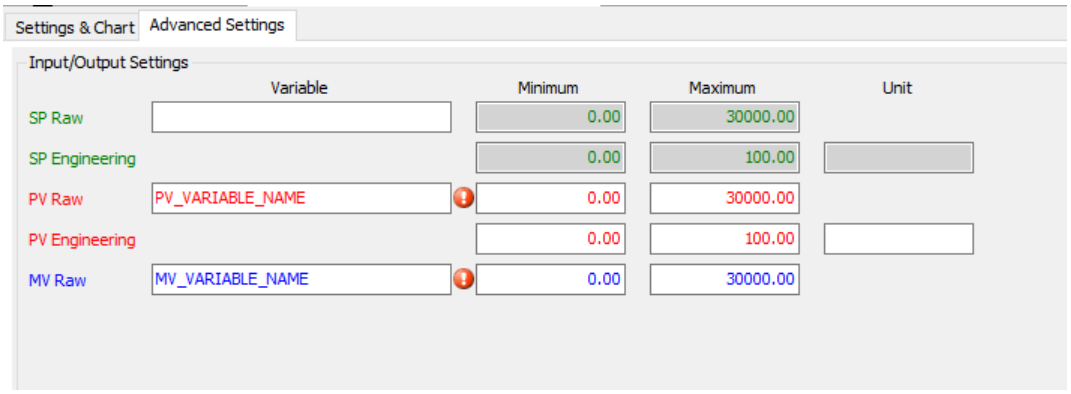

Figure 292: Group Input/Output Settings

In the *Variable* fields are configured the variables that are used as input (PV) and output (MV) of PID. The *PID Control* object only accepts variables of REAL type.

The SP is the only variable field that can be left blank. In this case, it is assumed that an internal variable should be used for this field. These variables must be declared global variables in other objects of the application as field networks configuration or *GVL Objects*. The use of external variables in this field allows, for example, cascading control strategies with the *PID Control* object.

The *Minimum* and *Maximum* fields define the operating range of the variables SP, PV and MV. The *Minimum SP* and *Maximum SP* fields, do not allow editing. These fields assume the values of the fields *PV Engineering Minimum* and *PV Engineering Maximum*, respectively.

The correct adjustment of this information is of great importance for the proper functioning of the PID loop. It is important to note that these values are also used to validate data entry fields of *Online Settings* group of *Setting & Chart* tab, according to the parameter setting *View Options*.

8.19.2.2.2. Group: Control Settings

This group allows you to configure some parameters related to the operating mode of the PID controller:

- *Sample Time (ms)*: Set the time interval that the PID is running, may vary from 1 ms to 1,000,000 ms.
- *Control*: This input parameter selects the right direction of action MV to provide a negative feedback. If a wrong selection is made, the resulting feedback is positive, and the PID will not be able to control the process. *Direct* control should be selected when MV should increase in response to an increase in PV. *Reverse* control should be selected when MV is expected to decrease in response to an increase in PV.
- *Enable...* : These fields enable individually the four actions (proportional, integrative, derivative and derivative in PV) that make up the block PID.

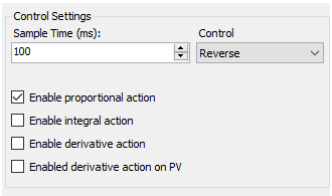

Figure 293: Group: Control Settings

8.19.2.2.3. Tab: Project Settings

This tab contains the option *Automatic Task Association*, with the option enabled the controller is automatically associated to a task system, which allows it to be used normally. If the option is disabled, the controller should be associated with any task manually or called in some POU user.

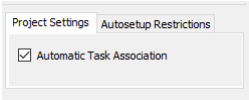

Figure 294: Tab: Project Settings

8.19.2.2.4. Tab: Autosetup Restrictions

This tab contains fields that define the minimum and maximum values that the auto-tuning can be assigned to the variables MV and PV.

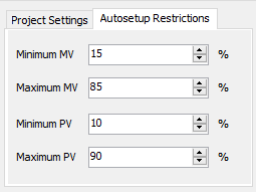

Figure 295: Autosetup Restrictions

8.19.2.3. Auto-Tuning Procedure

The procedure of automatic tuning of *PID Control* is accomplished by accessing object window *Autosetup*. This is accomplished by clicking on the button *Autosetup* located in the *Online Settings* group of *Settings & Chart* tab. The figures below displays the windows from *Autosetup*.

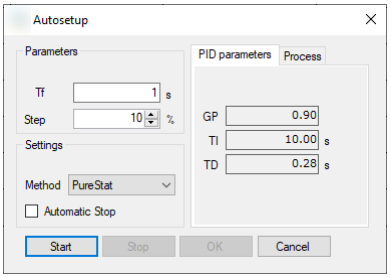

Figure 296: Autosetup, PID Parameters

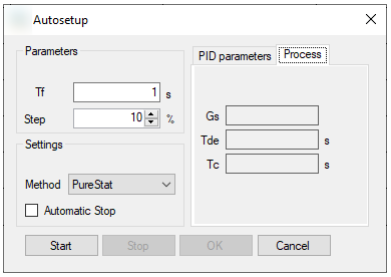

Figure 297: Autosetup, Process

Through the window *Autosetup* it is possible perform the procedure of tuning the PID controller parameters using the method of synthesis. For the application of this method, it is necessary to be known the parameters of a first-order system that represents the process. In this way, it is necessary that these parameters are identified by conducting an experiment in open loop, controlled by an operator. This experiment consists of applying a step signal to the process and wait for the steady state response. See below the necessary steps to perform the auto tuning procedure.

In the *Tf* field is configured the time constant in desired closed loop. It is important to emphasize that the synthesis method does not produce good results when $Tf < Tc < 10$ and when $Tc / (Tde + SampleTime/2) < 10$. Where *Tc* and *Tde* correspond to the time constant and the transport delay of the process and *SampleTime* is the range of application of PID.

On the *Step* field you can configure the step (in percentage) that is applied to the process in the experiment. In other words, the signal applied at the output of the controller will be corresponding to (MV + MV*Step).

In the field *Method* the user can select between two methods to perform the identification of process parameters. The first of them *PureStat*, uses only statistical information, while the second *PolyStat* uses statistical information in conjunction with polynomial approximation. The statistical information is used to try to soften the presence of noise in the process.

The *Automatic Stop* option enables the automatic termination of the experiment required to obtain the approximate parameters of a first-order model to be used by the method of synthesis. To determine the end the algorithm monitors the signal PV waiting for its stability. To wait for the stability the algorithm monitors the signal to each sampling. If the shifting from one sample to another is less than 2% of the total variation the algorithm waits 800 samples within the range to consider the signal stable. If it is perceived by graphical representation that the signal does not stabilize the experiment can be toggled to manual mode.

The fields *GP*, *TI* and *TD* of the PID parameters tab, express, initially, the current values configured for the controller parameters.

The fields *Gs*, *Tc* and *Tde* of the Process tab, express values of the static gain of the process, the time constant of the process and the deadtime of the process, respectively.

Before you start the experiment of identification of process parameters and tuning of the parameters of the controller, it is important that the process is in steady state. Guaranteed this situation the experiment can be started by pressing the *Start* button.

Having been the experiment started, the trend chart begins to monitor the process to ensure that the operator keep track of what is occurring. In the experiment, initially, the PID controller is passed automatically to manual operating mode.

Then, after a certain period of time the step signal is applied. It is important to note that the signal level is not applied immediately. In this interval in which the MV remains unchanged is being collected some statistical information that will be used to minimize the possible presence of noise in the process.

After applied the step, the process will begin to respond to this stimulus until steady state again. The experiment must be kept running with the process in steady state for certain period of time. The procedure can be stopped by pressing the *Stop* button. One should remember that being the *Automatic Stop* option is enabled at some point, the experiment will be terminated automatically by the tuning procedure. However, the operator is still able to terminate the experiment when convenient, even before the automatic finalizing.

When the *Stop* button is pressed the experiment is terminated and the new suggested parameters to the controller are displayed in *GP*, *TI* and *TD*. In order to send the parameters to the PLC the Write button must be pressed.

After the auto tuning procedure, the controller will remain working in its manual mode. To return to the automatic mode, press the *Auto* button of *Online Settings* group of *Settings & Chart* tab.

9. Installation

To install the MasterTool IEC XE development software, it is necessary to download the installation file from the site https://www.altus.com.br/suporte . If an instance of MasterTool IEC XE is already open and you only want to update it, the installer will ask you to close all open instances. After running the installer, follow the next steps to install the program:

9.1. Select Setup Language

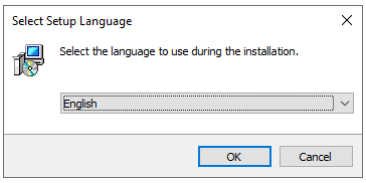

Figure 298: Select Setup Language

Select the installation language and press *OK* to continue with the process. This action will trigger the MasterTool IEC XE installation.

9.2. License Agreement

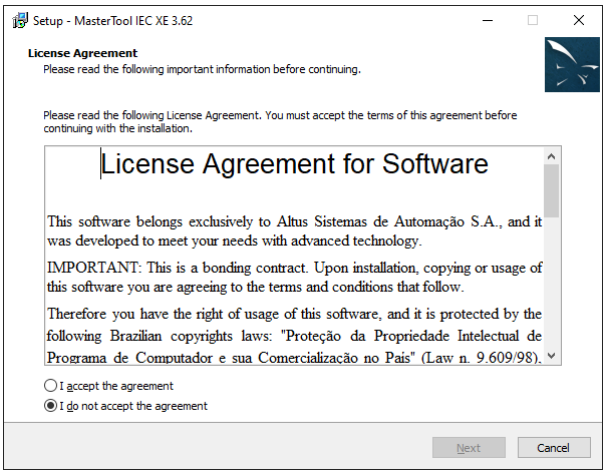

Figure 299: License Agreement

This is the license contract, where the user needs to read and accept to proceed with the installation. If the user click in the *Cancel* button, the installation will be canceled. After accepting the contract, it's necessary to click in the *Next* button to advance to the next step of the installer. This is valid for every installer screen that has a *Next* button.

9.3. Select Components

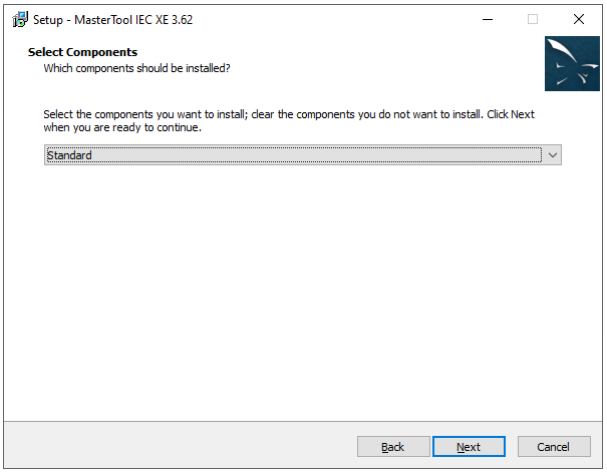

Figure 300: Select Components Standard

This is screen is where the user select which components will be installed in MasterTool IEC XE . The *Standard* install only the default components needed for MasterTool IEC XE , but there's also the *Extended* option, where some more packages can be installed.

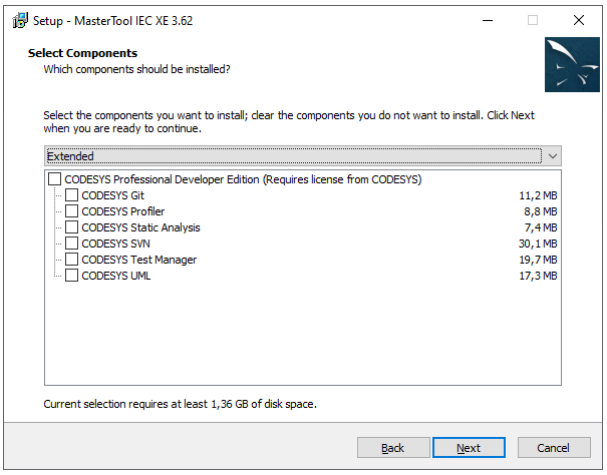

Figure 301: Select Components Extended

Note: In this page there's also a **Back** button. If the user click in this button, it will return to the right back page of the installer. It is valid for every installer page that has an **Back** button.

9.4. Select Additional Tasks

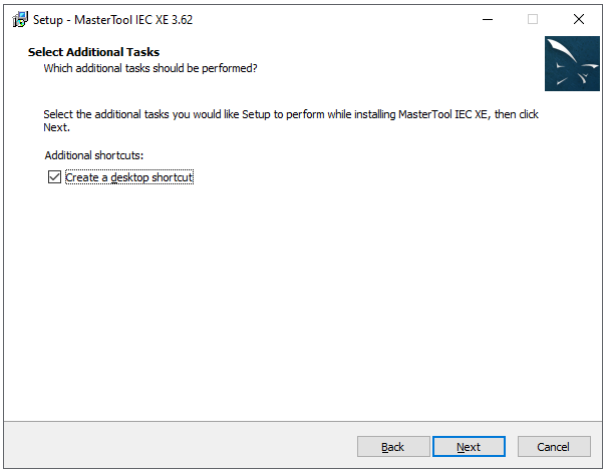

Figure 302: Select Additional Tasks

The in this case, the user can select additional tasks to be done while MasterTool IEC XE is installed.

9.5. Ready to Install

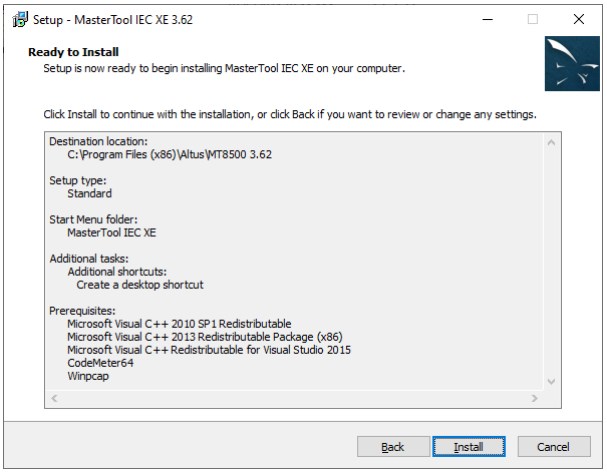

Figure 303: Ready to Install

This is the last page before the it start the installation in fact. It shows some informations like *Destination location*, *Setup type*, the select *Additional Tasks* and the necessary *Prerequisites*. After clicking in *Install*, it will start the installation, but in this page it's already possible to click in the **Back** button and change any option selected. Also, there's the option to click in the *Cancel* button and cancel the entire operation.

9.6. Installing

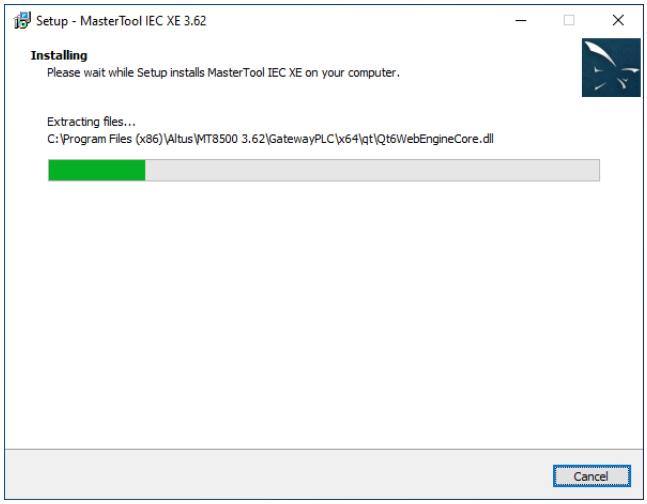

Figure 304: Installing

This is where MasterTool IEC XE is in fact installing. Creating folders, extracting and moving files, installing assemblies, loading plugins chaches, etc. Still there's the option to cancel the operation by clicking in the *Cancel* button.

9.7. Completing the MasterTool IEC XE Setup Wizard

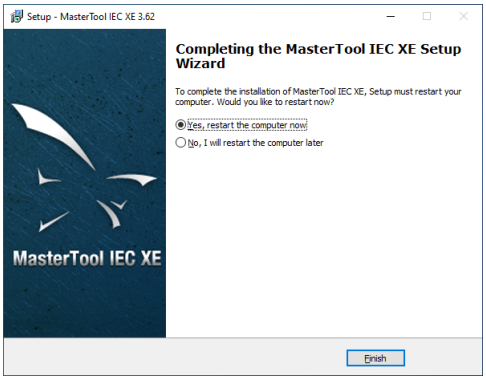

Figure 305: Completing the MasterTool IEC XE Setup Wizard

This is the last page of the installation. It will inform installation was complete and will show the user this two options:

- *Yes, restart my computer now*: Will restart the computer at that moment.
- *No, I will restart the computer later*: Will close the installer, so the user will choose when will restart the computer.

10. Diagnostics

10.1. Diagnostics

Each project has their diagnostics objects (located in the devices tree), like *System_Diagnostics*, inside *SystemGVLs* folder.

The diagnostics are automatically updated. For example, if an module is added to the project, their diagnostics will be added to *Module_Diagnostics*. The same applies for when a device is removed.

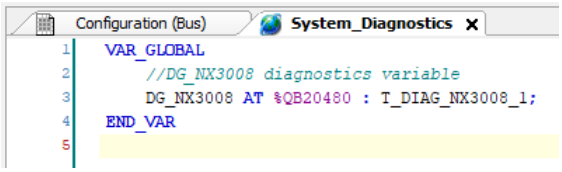

Figure 306: System Diagnostics

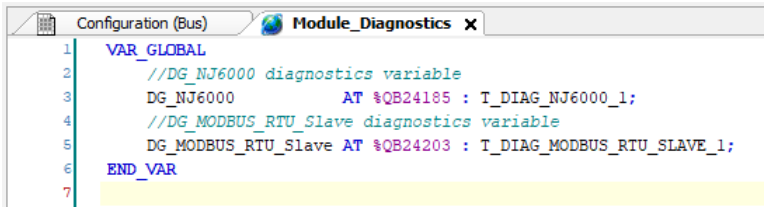

Figure 307: Module Diagnostics

Two other objects are created in the devices tree for Symbolic Mapping communication drivers. They are named *Disables* and *ReqDiagnostics*.