



MasterTool IEC XE

User Manual

MT8500

Rev. C 10/2014
Doc. Code.: MU299609



altus

www.altus.com.br



No part of this document may be copied or reproduced in any form without the prior written consent of Altus Sistemas de Automação S.A. who reserves the right to carry out alterations without prior advice.

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.

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.

COPYRIGHTS

Nexto and MasterTool IEC XE are registered trademarks of Altus Sistemas de Automação S.A.

Windows is a registered trademark of Microsoft Corporation.

Summary

SUMMARY	II
1. INTRODUCTION	9
Documents Related to this Manual	9
General Regards on ALTUS Documentation.....	9
MasterTool IEC XE Support Documentation	10
Visual Inspection	10
Technical Support	10
Warning Messages Used in this Manual	11
2. TECHNICAL DESCRIPTION	12
MasterTool IEC XE Versions.....	12
Languages.....	13
Compatibility with Other Product.....	13
Minimum Requirements	13
Data for Purchase.....	14
Included Items	14
Product Code.....	14
3. CONCEPTS AND BASIC COMPONENTS.....	15
Introduction	15
Basic Concepts	15
Advanced Functionalities.....	15
Object Orientation in Programming and Project Structure	15
Special Data Types.....	16
Operators and Special Variables	16
New User Management and Access Rights Concept	16
Features in Editors.....	16
Library Versions.....	16
Additional Functionalities	16
Profiles.....	17
Single.....	17
Basic.....	18
Normal.....	18
Expert	19
Custom	19
General Table	20
Project.....	21
Project Template	21
POUs Window.....	21
Project Settings and Project Information	21
External File	22
Device, Device Tree	22

Generals.....	23
Installing of Devices on the Local System.....	24
Arranging and Configuring Objects in the Devices Tree - Rules	24
Application.....	25
Task Configuration	25
Important Notes for Multitasking Systems	25
Code Generation and Online Change.....	26
Code Generation and Compile Information	26
Online Change.....	26
Boot Application (Boot Project)	28
Projects Download/Login Method without Project Differences	28
Monitoring	29
Debugging.....	29
Breakpoints	29
Stepping.....	29
Printing.....	30
Security.....	30
Project	30
PLC	30
4. QUICK START	31
Project Profiles	31
Start MasterTool IEC XE.....	31
Adding Modules.....	34
Creating POU.....	36
Creating Tasks.....	38
Task Configuration.....	40
POU – Task Connection.....	41
Maximum Number of Tasks	42
CPU Configuration.....	42
Libraries	44
Inserting a Protocol Instance.....	45
MODBUS RTU.....	45
MODBUS Ethernet	47
Finding the Network	49
Building a Project.....	51
Login.....	52
Run Mode.....	54
Stop Mode	55
Logout.....	56
Simulation Mode.....	57
Project Upload	58
Create and Run Projects.....	60
Declare Variables in MainPrg.....	60
Enter Programming Code in the Body of MainPrg.....	61
Create a Programming POU (ST function block FB1)	62
Define the Resources for Running and Controlling the Program on Nexto.....	63
Run and Watch the Application on Nexto.....	63
Debug an Application	66
Help.....	66
Context Sensitive Help.....	66
Uninstallation, Update, Repair.....	67
5. USER INTERFACE.....	68

User Interface Components	68
Windows, Views, Editor Windows	69
Customizing User Interface	70
Arranging Menu Bars and Tool Bars	70
Zoom	70
User Interface in Online Mode	70
Standard Menus and Commands.....	72
Standard Menus and Commands.....	72
User Files Memory	77
 6. USER AND ACCESS RIGHT MANAGEMENT.....	 81
User Management.....	81
Access Right Management.....	81
Users and Groups	82
Access Rights	85
Status	87
Information.....	87
 7. MENU COMMANDS	 88
File Menu.....	88
New Project.....	88
Open Project.....	90
Close Project	91
Save Project.....	91
Save Project As	92
Project Archive: Extract Archive... ..	92
Project Archive: Save/Send Archive.....	94
Source Upload.....	96
Source Download	97
Print	97
Page Setup.....	97
Recent Projects.....	97
Exit	98
Edit Menu.....	98
Undo/Redo	98
Clipboard.....	99
Select All.....	100
Find/Replace	100
Browse.....	104
Insert File As Text.....	105
Advanced.....	105
Bookmarks	106
Input Assistant.....	107
Auto Declare... ..	108
Messages View.....	110
View Menu	110
Default Navigators	111
Messages	111
Element Properties	112
Product Library	112
Toolbox	113
Watch List View	113
Watch	113
Breakpoints	113

Call Stack	114
Cross Reference View	115
Start Page	116
Full Screen	116
Properties	116
Project Menu.....	123
Add Object	124
Add Device.....	124
Add Folder	125
Edit Object	126
Edit Object with	126
Project Information	126
Project Settings.....	131
Project Update.....	140
Export to CSV	142
Import from CSV	143
Document	144
Compare	145
User Management	147
Menus from the Editors of Programming Languages	150
FBD/LD/IL Editor Commands.....	151
CFC Commands	167
SFC Commands	175
Build Menu.....	182
Generate Code.....	182
Clean	182
Clean All	182
Online Menu	183
Login	183
Logout	185
Create boot application.....	185
Logoff current online user.....	185
Download	185
Online Change.....	186
Source download to connected device	186
Configuration of Basic Cluster.....	187
Reset Warm.....	187
Reset Cold.....	187
Reset Origin	187
Simulation	187
Debug Menu.....	188
Start	189
Stop	189
Breakpoints	189
New Breakpoint	189
Toggle Breakpoint.....	192
Step Over.....	192
Step Into	192
Step Out.....	193
Run to Cursor	193
Set Next Statement.....	193
Show Next Statement.....	194
Write Values.....	194
Force Values.....	194
Unforce Values.....	197
Add All Forces to the Watchlist.....	198

Display Mode	198
Tools Menu.....	198
Library Repository	198
Install Library	202
Device Repository	203
Install Device	204
Options... ..	204
Window Menu.....	225
Next Editor	226
Previous Editor.....	226
Close Editor.....	226
Close All Editors	226
Reset Window Layout.....	226
New Horizontal Tab Group.....	226
New Vertical Tab Group	227
Float.....	227
Dock	227
Auto Hide	227
Next Pane	227
Previous Pane	227
Window <n>	227
Windows... ..	227
Help Menu.....	228
Contents.....	228
Index.....	228
Search.....	228
Contact Support.....	229
Update Software License	229
Documentation	229
Altus Home Page.....	229
About.....	229
Bus Editor Menu	229
Zoom out	230
Zoom in	230
Rulers	230
Ports.....	230
Grid	230
Declaration.....	230
Tabular Declaration Editor - Declaration.....	230
Edit Declaration Header	230
Insert.....	230
Move Down.....	230
Move Up.....	230
Trace	231
8. EDITORS.....	232
General Regards on Editors	232
Bus Editor	232
Add Device.....	233
Remove Device	233
Digital I/O Module Editor.....	233
Process Data	233
Module Parameters.....	234
Bus: I/O Mapping.....	235
Bill of Materials	236

Configuration and Consumption	237
Programming Language Editors.....	238
Declaration Editor	238
Textual Declaration Editor	238
Tabular Declaration Editor	239
Declaration Editor in Online Mode	241
Device Editor.....	242
Communication Settings	242
Files	248
Log	249
DUT Editor	250
FBD/LD/IL Editor	251
Global Variables List Editor	251
Library Manager Editor	251
Library Manager Editor Window	252
Libraries Menu	254
Task Editor	255
Task Configuration.....	255
Task Editor, Usage	255
Task Editor in Online Mode.....	259
Task Configuration Commands	261
Trace Editor	261
Trace.....	261
Shortcuts for Trace	264
Trace Editor.....	265
Add Variable	268
Configuration	269
Appearance.....	275
Trace Editor in Online Mode	276
Watch List Editor	277
Watch View / Watch List Editor	277
Create Watchlist	278
Watch List in Online Mode	278
MODBUS Editor	279
MODBUS RTU Master for Direct Representation (%Q)	280
MODBUS RTU Master for Symbolic Mapping	283
MODBUS RTU Slave for Direct Representation (%Q)	286
MODBUS RTU Slave for Symbolic Mapping.....	287
MODBUS Ethernet Client for Direct Representation (%Q)	289
MODBUS Ethernet Client for Symbolic Mapping	292
MODBUS Ethernet Client for Direct Representation (%Q)	296
MODBUS Ethernet Server por Symbolic Mapping.....	299
PROFIBUS Editor.....	300
CPU Editor.....	300
Serial Interfaces.....	301
Ethernet Interfaces.....	302
PID Control Editor.....	303
Insert PID Control object in the application.....	303
Graphical Environment	304
9. INSTALLATION	317
10.DIAGNOSTICS	324
Diagnostics Page	324

Diagnostics 324

11.GLOSSARY 328

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 six programming languages defined by IEC 61131-3 standard: Structured Text (ST), Sequential Function Chart (SFC), Function Block Diagram (FBD), Ladder Diagram (LD), Instruction List (IL) 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.

Documents Related to this Manual

For additional information about MasterTool IEC XE, the user can examine other specific documents in addition to this one. These documents are available in its last review on www.altus.com.br.

General Regards on ALTUS Documentation

Each product has a document called Technical Characteristics (CT), where there are the characteristics for the product in question. Additionally, the product may have User Manuals (manual's codes, if applicable, are always mentioned at CTs from the respective modules).

MasterTool IEC XE Support Documentation

It is advisable to consult the following documents as a source of additional information:

Code	Description	Language
CE114000 CT114000 CS114000	Nexto Series – Features and Configuration Série Nexto – Características e Configurações Serie Nexto – Especificaciones y Configuraciones	English Portuguese Spanish
CE114100 CT114100 CS114100	CPUs Nexto Series – Features and Configuration CPUs Série Nexto – Características e Configurações CPUs Serie Nexto – Especificaciones y Configuraciones	English Portuguese Spanish
CE103705 CT103705 CS103705	MasterTool IEC XE – Features and Configuration MasterTool IEC XE – Características e Configurações MasterTool IEC XE – Especificaciones y Configuraciones	English Portuguese Spanish
MU214600 MU214000 MU214300	Nexto Series User Manual Manual de Utilização Série Nexto Manual Del Usuario Serie Nexto	English Portuguese Spanish
MU214605 MU214100 MU214305	Nexto Séries CPUs User Manual de Utilização CPUs Série Nexto Manual del Usuario CPUs Serie Nexto	English Portuguese Spanish
MU299609 MU299048 MU299800	MasterTool IEC XE User Manual Manual de Utilização MasterTool IEC XE Manual del Usuario MasterTool IEC XE	English Portuguese Spanish
MP399609 MP399048 MP399800	IEC 61131 Programming Manual Manual de Programação IEC 61131 Manual de Programación IEC 61131	English Portuguese Spanish

Table 1-1. Support Documentation

Visual Inspection

Prior to installation, we recommend performing a careful visual inspection of equipment, by checking if there is damage caused by shipping. Make sure all components of your order are in perfect condition. In case of defects, inform the transportation company and the nearest Altus representative or distributor.

CAUTION:

Before removing modules from the package, it is important to discharge eventual static potentials accrued in the body. For this, touch (with nude hands) in a metallic surface grounded before modules handling. Such procedure ensures that the levels of static electricity supported by the module will not be overcome.

It is important to record the serial number of each item received, as well as software revisions, if any. This information will be necessary if you need to contact Altus Technical Support.

Technical Support

To contact Altus Technical Support in São Leopoldo, RS, call +55 51 3589-9500. To find the existent centers of Altus Technical Support in other locations, see our website (www.altus.com.br) or send an email to altus@altus.com.br.

If the equipment is already installed, please have the following information when requesting assistance:

- Models of equipment used and the configuration of installed system
- Serial number of CPU
- Equipment review and executive software version, listed on the label affixed to the product side
- Information about the operation of CPU, obtained through MasterTool IEC XE programmer, and graphical display from CPU
- Contents of the application program, obtained through MasterTool IEC XE programmer

- Version of the programmer used

Warning Messages Used in this Manual

In this manual, warning messages will present the following formats and meanings:

DANGER:

Relates potential causes, which if not noted, generate damages to physical integrity and health, property, environment and production loss.

CAUTION:

Relates configuration details, application and installation that shall be followed to avoid condition that could lead to system fail, and its related consequences.

ATTENTION:

Indicate important details to configuration, application or installation to obtain the maximum operation performance from the system.

2. Technical Description

MasterTool IEC XE Versions

MasterTool IEC XE MT8500 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 on Nexto bus and only NX3010 on CPUs
- **Basic:** software that allows the programming and project loading of up to 2048 I/O points on CPUs NX3010 and NX3020
- **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	6	6	6	6
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
Instruction List (IL)	Yes	Yes	Yes	Yes
Continuous Function Chart (CFC)	Yes	Yes	Yes	Yes
Supported Nexto Series CPUs	1	2	3	3
NX3010	Yes	Yes	Yes	Yes
NX3020	No	Yes	Yes	Yes
NX3030	No	No	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
Fieldbus support	No	Yes	Yes	Yes
Fieldbus redundancy support	No	No	Yes	Yes
Half-Cluster redundancy support	No	No	No	Yes
Limitation of number of I/O points	Yes	Yes	No	No
Maximum number of I/O points	320	2048	Unlimited	Unlimited

Table 2-1. MasterTool IEC XE Versions Features

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

NOTE: Maximum number of I/O: To MasterTool IEC XE Professional and Advanced versions there is no limit for maximum number of I/O points. In this case, MT8500 makes no checking. However, the amount of points is limited by the memory occupation of %I and Q% in each of the CPU models.

ATTENTION:

The programming of Nexto Series CPs is available from MasterTool IEC XE version 1.00.

Languages

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

Compatibility with Other Product

Earlier to 1.29 versions of MasterTool IEC XE are not compatible with all revisions of products of Nexto Series modules. With any of the products related below (either in its software version or product revision) an updating to 1.29 or higher version is required.

	Software Version	Product Revision
NX1001	1.1.0.1 or higher	AB or higher
NX1005	1.1.0.1 or higher	AB or higher
NX2001	1.0.1.1 or higher	AB or higher
NX2020	1.0.1.1 or higher	AC or higher
NX3010	1.2.1.0 or higher	AN or higher
NX3020	1.2.1.0 or higher	AH or higher
NX3030	1.2.1.0 or higher	AL or higher
NX4000	1.1.0.0 or higher	AD or higher
NX4010	1.0.1.0 or higher	AG or higher
NX5000	1.0.1.1 or higher	AJ or higher
NX5001	1.1.1.1 or higher	AM or higher
NX6000	1.0.1.2 or higher	AF or higher
NX6100	1.0.1.1 or higher	AE or higher

Table 2-2. Compatibility with Products Version 1.29

Earlier to 1.40 versions of MasterTool IEC XE do not allow the configuration of the MODBUS for Symbolic Mappings communication drivers. The same behavior is valid for the LibLogs Library use. In order to a project with these drivers or libraries may be charged in the CPU the products related below (either in its software version or product revision) need to be updated. In case it is not necessary, the use of MODBUS for Symbolic Mappings communication drivers or LibLogs Library there is no restriction for this version.

	Software Version	Product Revision
NX3010	1.3.0.22 or higher	AR or higher
NX3020	1.3.0.22 or higher	AL or higher
NX3030	1.3.0.22 or higher	AO or higher

Table 2-3. Compatibility with Products Version 1.40

Minimum Requirements

The following products are presented as minimum requirements for installation and use of MasterTool IEC XE, for any of its distributions versions:

	MasterTool IEC XE
Platform	PC with Windows XP® (32bits), Windows Vista® (32 bits) or Windows 7® (32bits or 64bits)
Processor	Intel Core 2 Duo 1.66 GHz (minimum)
Disc Space	1 Gbyte (minimum), 2 Gbytes (recommendable)
RAM Memory	2 Gbytes (minimum), 3 Gbytes (recommendable)
Resolution	1024 x 768 (recommendable)
Language	Any language

Table 2-4. Minimum Requirements

Data for Purchase

Included Items

The product may be acquired in two ways:

- Product package contain the following items:
 - MasterTool IEC XE software recorded on CDROM
 - Altus Software License Contract
- In a product package that contains the license for subsequent download of the software at Altus site: www.altus.com.br (Versions /Licenses)

Product Code

The following codes shall be used for product purchase:

Code	Denomination
MT8500 Lite	MasterTool IEC XE Lite
MT8500 Basic	MasterTool IEC XE Basic
MT8500 Basic /L	MasterTool IEC XE Basic/L (license)
MT8500 Professional	MasterTool IEC XE Professional
MT8500 Professional /L	MasterTool IEC XE Professional/L (license)
MT8500 Advanced	MasterTool IEC XE Advanced
MT8500 Advanced /L	MasterTool IEC XE Advanced /L (license)

Table 2-5. Codes for MasterTool IEC XE Purchase

3. Concepts and Basic Components

Introduction

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

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.
- **Standard and professional interface:** Predefined feature sets serve to be able to choose between a "standard" user interface with a reduced selection of features and less complexity and a "professional" environment supporting all features. When the programming system is initially started after the first installation on the system, the user will be prompted to choose one of the sets. Also later, the user still can switch the set and also a user-defined customization of the currently used feature set is possible. For the particular differences between the standard and professional version, please see **Features**.

Advanced Functionalities

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

Object Orientation in Programming and Project Structure

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

Applications are instances of independent programming objects.

Special Data Types

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

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

New User Management and Access Rights Concept

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

Features in Editors

- ST-Editor: Editing resources, Line break, Autocomplete, Inline Monitoring, and Inline set/reset-assignment.
- FBD, LD and IL are reversible and programmable in a combined editor.
- IL editor is a table editor.
- LD/FBD/IL editor: Main output in boxes with multiple outputs can be changed
- LD/FBD/IL editor: no automatic update of box parameters
- LD/FBD/IL 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

Library Versions

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

Additional Functionalities

- Configurable menus, toolbar and keyboard usage
- Customer-specific components might be plugged in
- 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
- Each object can be specified to be "internal" or "external" (late link in the runtime system)
- Precompile notes concerning syntax errors

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, 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

For each profile defined for RTS, MasterTool IEC XE can provide lots of compatible templates. When the user selects a template as model in the creation of a new project, the new application will be developed in accordance to a specific profile, adopting rules, features and patterns defined by the profile associated to the template. Each profile defines names for the tasks and programs, which will be previously created by the project templates. The developer is required to strictly follow the tasks name, although the suggested names for the respective programs may or not be followed. To ensure compatibility of a project to a certain profile throughout development, we use two approaches:

- MasterTool IEC XE only allows the creation of projects based on a template by selecting at the same time the profile to be used;
- In code generation, MasterTool IEC XE checks all the rules defined for the valid profile to the project.

The next sections detail the features or patterns of each profile, which follow a graduated scale of complexity. Based on these definitions, it is recommendable that the user always uses the simplest profile, which attends his applications needs. If the profile corresponding rules turns more into barriers than didactic simplifications, it is time to user go to a sophisticated profile. The programming tool allows the profile alteration of an existing project (see **Updating an Old Project**), but it is up to the developer make the necessary adjustments in order to the project gets compatible to the new selected profile.

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

Single

In the Single project profile, the application has only the MainTask user task. This task is responsible for the execution of a single Program unit called MainPrg. This single program can call other programming units from the Program, Function or Function Block type, but all user code will be executed exclusively by the MainTask.

In this profile, the MainTask is cyclic with priority fixed in 13 (thirteen) and executes exclusively the MainPrg program. The MainTask is already completely defined and the developer must create the MainPrg program using any of the IEC 61131-3 languages. It is not always possible to convert a program to another language, but it is always possible to create a new program with the same name constructed with a different language. The MasterTool IEC XE standard option is to use the

MasterTool Standard Project associated to the Single profile, which also includes the MainPrg program created in the selected language when the project was created.

This type of application never needs to consider issues as data consistence, resource sharing or mutual exclusion mechanisms.

Task	POU	Priority	Type	Cycle Time	Event
MainTask	MainPrg	13	Cyclic	100 ms	-

Table 3-1. Task on Single Profile

Basic

In the Basic project profile, the application has a user task from the Continuous type called MainTask, which executes the program in a continuous loop (with no definition of cycle time) with priority fixed in 13. This task is responsible for the execution of a single programming unit POU called MainPrg. It is important to stress that the cycle time can vary according to the communication task quantity used, as in this mode, the main task is interrupted by communication tasks.

This profile allows the inclusion of two interruption tasks with higher priority which can preempt the MainTask at any moment: the task called ExternInterruptTask00 is an interruption task from the Extern type with priority fixed in 02 (two); the task called TimeInterruptTask00 is an interruption task from the cyclic type with priority fixed in 1 (one).

The basic project model includes these three task already completely defined, as shown in Table 3-2. The developer needs only to implement the program contents.

Task	POU	Priority	Type	Cycle Time	Event
MainTask	MainPrg	13	Continuous	-	-
ExternInterruptTask00	ExternInterruptPrg00	02	External	-	IO_EVT_0
TimeInterruptTask00	TimeInterruptTask00	01	Cyclic	20 ms	-

Table 3-2. Tasks on Basic Profile

Normal

In the Normal profile, the application has a user task from the cyclic type called MainTask. This task is responsible for the execution of a single programming unit POU called MainPrg. This program and this task are similar to the only task and only program of the Single profile, but here the application can integrate additional user tasks. These other tasks are called CyclicTask00, CyclicTask01, etc., each one responsible for the execution of the respective program CyclicPrg<nn>. The CyclicPrg<nn> tasks are always from the Cyclic type with priority fixed in 13 (thirteen), equal to MainTask. These two types form a group called basic tasks, which the associated programs can call other POUs from the Program, function or Function Block types.

Furthermore, this profile can include interruption tasks with higher priority than the basic tasks, which can preempt these tasks execution at any time.

The task called ExternInterruptTask00 is an interruption task from the type Extern which the execution is triggered by some external event, as control signal variation on a serial port or a digital input variation on the NEXTO bus. This task priority is fixed in 02 (two), and it is responsible exclusively for the execution of the ExternInterruptPrg00 program. The task called TimeInterruptTask00 is an interruption task from the Cyclic type with priority fixed in 01 (one), and it is responsible for the execution exclusively of the TimeInterruptPrg00 program.

In the Normal project model, there are five tasks already completely defined as shown on Table 3-3. The developer needs to implement the program contents choosing any language from the IEC 61131-3 standard. The intervals and task trigger events can be configured by the developer and the unnecessary task can be eliminated.

Task	POU	Priority	Type	Cycle Time	Event
MainTask	MainPrg	13	Cyclic	100 ms	-
CyclicTask00	CyclicPrg00	13	Cyclic	200 ms	-
CyclicTask01	CyclicPrg01	13	Cyclic	500 ms	-
ExternInterruptTask00	ExternInterruptPrg00	02	External	-	IO_EVT_0
TimeInterruptTask00	TimeInterruptTask00	01	Cyclic	20 ms	-

Table 3-3. Tasks on Normal Profile

Expert

The Expert project profile includes the same basic tasks MainTask and CyclicTask<nn> and interruption tasks ExternInterruptTask00 and TimeInterruptTask00 with the same priorities (13; 02 and 01 respectively), but it is an expansion comparing with the previous profiles, as it allows multiple interruption tasks. Therefore, the application can include several ExternInterruptTask<nn> or TimeInterruptTask<nn> tasks executing the ExternInterruptPrg<nn> and TimeInterruptPrg<nn> programs. The interruption tasks additional priorities can be freely selected from 08 to 12. In this profile, besides the standard programs, each task can execute additional programs.

In this project profile, the application can also include the user task FreeTask from the Freewheeling task, with priority 31, responsible for the FreePrg program execution. As this task is low priority it can be interrupted by all others, and it can execute codes that might be blocked.

There are eight tasks (and it's POU's) already completely defined, as shown on Table 3-4, as the respective programs associated to the chosen language. The intervals and trigger events of any task and the interruption tasks priorities can be configured by the user.

At the application development using the Expert project profile, a special care is necessary with the interruption task scaling. In case there is information and resources sharing among these tasks or between these tasks and the basic tasks it is strongly recommended to adopt strategies to guarantee the data consistence.

Task	POU	Priority	Type	Cycle Time	Event
MainTask	MainPrg	13	Cyclic	100 ms	-
CyclicTask00	CyclicPrg00	13	Cyclic	200 ms	-
CyclicTask01	CyclicPrg01	13	Cyclic	500 ms	-
ExternInterruptTask00	ExternInterruptPrg00	02	External	-	IO_EVT_0
TimeInterruptTask00	TimeInterruptTask00	01	Cyclic	20 ms	-
ExternInterruptTask01	ExternInterruptPrg01	11	External	-	IO_EVT_1
TimeInterruptTask01	TimeInterruptPrg01	09	Cyclic	30 ms	-
FreeTask	FreePrg	31	Continuous	-	-

Table 3-4. Tasks on Expert Profile

Custom

The Custom project profile allows the developer to explore all Runtime System potentialities, implemented in the Nexto Series processing centrals. No functionality is disabled; no priority, task

and programs association or nomenclature is imposed. The only exception is for MainTask, which should always have this name in this Profile.

Besides the real time tasks with 00 to 15 priorities, which are scaled by priority, this profile also allows the tasks definition with smaller priorities smaller from 16 to 31. In this range it is used the Completely Fair Scheduler (time-sharing), which is necessary for the execution of the codes that might be blocked (e.g. the use of sockets).

The developer is free to partially follow or not the organization defined in the other project profiles, according to the particularities of his application. On the other hand, the custom model associated to this profile does not need pre-defined elements (task, program or parameter). The creation of all application elements is up to the developer. However, the user can generate the same elements available for the Expert profile.

Task	POU	Priority	Type	Cycle Time	Event
MainTask	MainPrg	13	Cyclic	100 ms	-
CyclicTask00	CyclicPrg00	13	Cyclic	200 ms	-
CyclicTask01	CyclicPrg01	13	Cyclic	500 ms	-
ExternInterruptTask00	ExternInterruptPrg00	02	External		IO_EVT_0
TimeInterruptTask00	TimeInterruptTask00	01	Cyclic	20 ms	-
ExternInterruptTask01	ExternInterruptPrg01	11	External	-	IO_EVT_1
TimeInterruptTask01	TimeInterruptPrg01	09	Cyclic	30 ms	-
FreeTask	FreePrg	31	Continuous	-	-

Table 3-5. Tasks on Custom Profile

General Table

		Project Profiles				
Checks		Single	Basic	Normal	Expert	Custom
Total of tasks	Quantity	01	[01..03]	[01..32]	[01..32]	[01..32]
Programs per task	Quantity	01	01	01	<n>	<n>
Main Task	Name	MainTask	MainTask	MainTask	MainTask	MainTask
	Type	Cyclic	Continuous	Cyclic	Cyclic	<n>
	Priority	13	13	13	13	<n>
	Quantity	01	01	01	01	01
Time Interrupt Task	Name		TimeInterrupt Task00	TimeInterrupt Task00	TimeInterrupt Task<nn>	<n>
	Type		Cyclic	Cyclic	Cyclic	Cyclic
	Priority		01	01	01 or [08..12]	<n>
	Quantity		[00..01]	[00..01]	[00..31]	[00..31]
Extern Interrupt Task	Name		ExternInterruptTask00	ExternInterruptTask00	ExternInterruptTask<nn>	<n>
	Type		External	External	External	External
	Priority		02	02	02 or [08..12]	<n>
	Quantity		[00..01]	[00..01]	[00..31]	[00..31]
Cyclic Task	Name			CyclicTask<nn>	CyclicTask<n>	<n>
	Type			Cyclic	Cyclic	Cyclic
	Priority			13	13	<n>
	Quantity			[00..31]	[00..31]	[00..31]
Free Task	Name				FreeTask	<n>
	Type				Freewheeling	Freewheeling
	Priority				31	<n>

Event Task	Quantity				[00..01]	[00..01]
	Name					<n>
	Type					Event
	Priority					<n>
	Quantity					[00..31]

Table 3-6. General Table Profile x Task

NOTE: The suggested names for the POU's associated to the tasks are not consisted in any profile. They can be replaced since they are also replaced in the tasks configurations.

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 POU's 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.

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**.

POUs Window

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

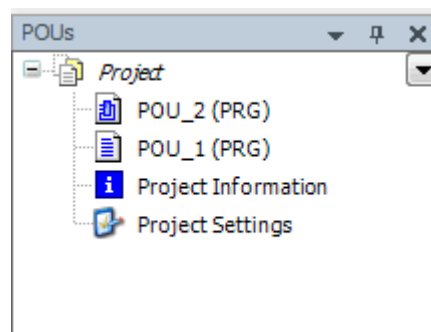


Figure 3-1. POU's Window

Project Settings and Project Information

The Project Settings and Project Information, per default are objects in the POU's 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** .

External File

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.

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 3-2). 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.

Generals

See in Figure 3-2 the “rules” for inserting objects in the device tree:

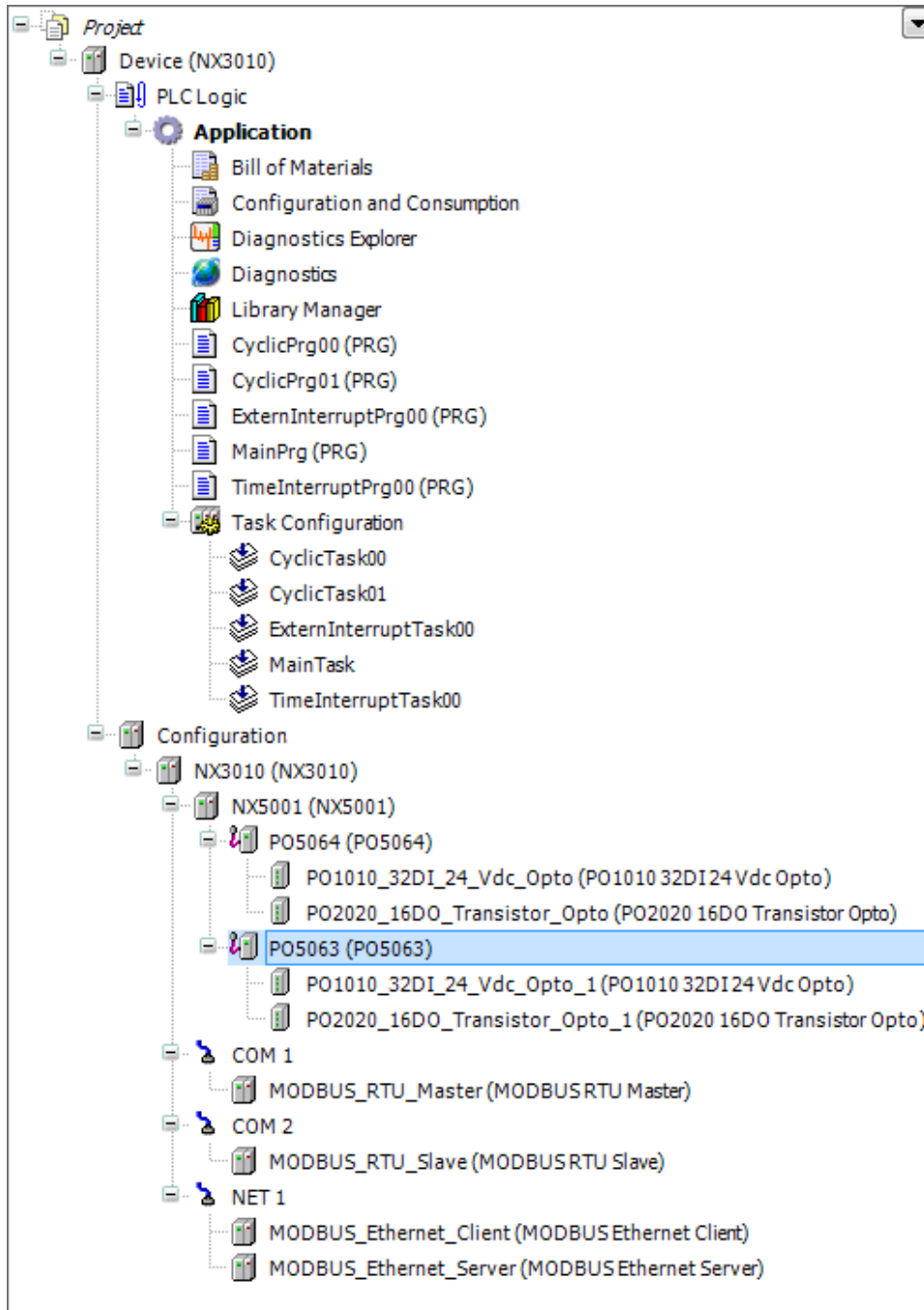


Figure 3-2. Device Tree

- 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
	Device connected
	Device not connected, not configured, with errors or in de simulation mode
	Device connected with diagnosis
	Device configured but not connected
	PLC is stopped and/or de I/O updating while PLC is stopped

Table 3-7. Device Status in Online Mode

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 **Install Library**). 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.

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 projects' device tree in the project.

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 POU's 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 POU's 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 the *MasterTool 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 POU's). Additionally it might have assigned resource objects like Global Variables Lists, Libraries etc., which - in contrast to those managed in the POU's window - only can be used by the particular application and its “childs”.

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.

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 *MasterTool 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 POU's 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.

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.

Code Generation and Online Change

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.

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 from 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.

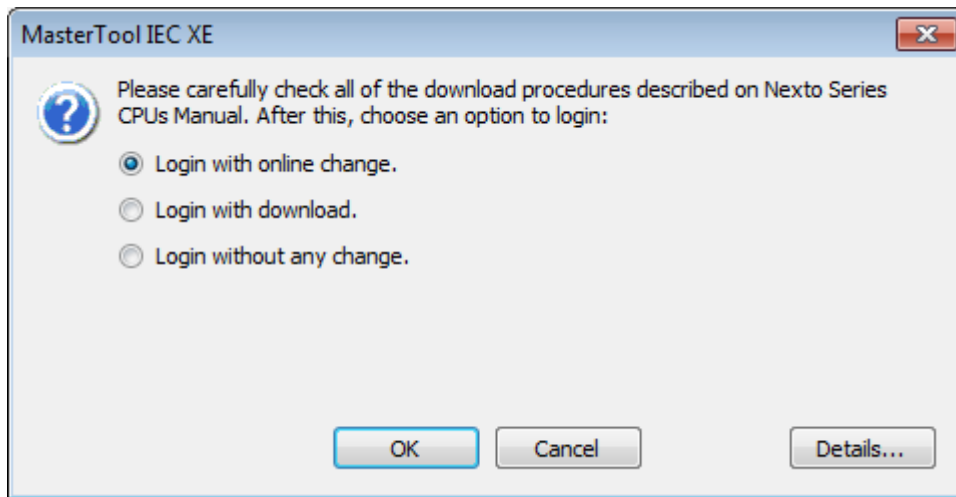


Figure 3-3. Login Dialog

- *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.

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:

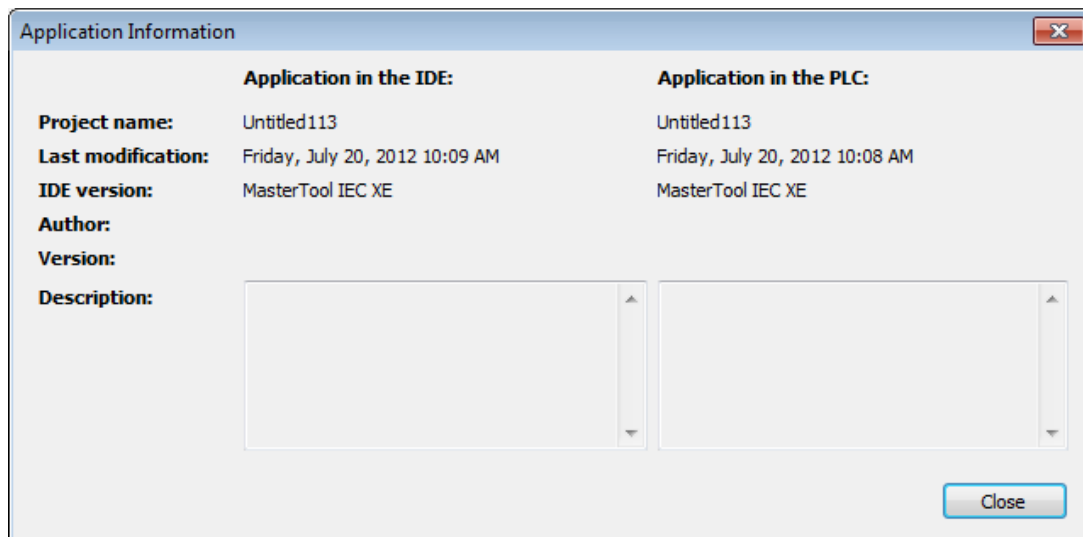


Figure 3-4. Application Information Dialog

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.

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.

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** .

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 **do Declaration**.
- 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 Editor** 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.

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.

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 **Breakpoint** for a description of the commands concerning breakpoints.

Stepping

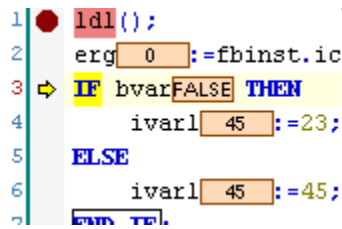
Stepping allows a controlled execution of an application program, e.g. for debugging purposes. Basically you step from one instruction to the next one by repeated use of the key <F10>, but also the user can step over POUs which are called.

- 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 **Breakpoint** 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.



```
1 ldl();
2 erg 0 :=fbinst.ic
3 IF bvarFALSE THEN
4   ivarl 45 :=23;
5 ELSE
6   ivarl 45 :=45;
7 END IF.
```

Figure 3-5. Step Into

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. For further information, see **Export to CSV**.

Security

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.

PLC

Depending on the device there might be an access control related to objects and files in the PLC. See **Users and Groups** 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.

The following items will be explored:

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

Project Profiles

A project profile in the MasterTool IEC XE is a group of rules, common features and patterns used in an industrial automation solution development, a profile which implies in the application implementation form. With the diversity of application types supported by the Nexto Series Runtime System, following a profile is a way to simplify the programming complexity.

The applications can be created according to the following profiles:

- Single
- Basic
- Normal
- Expert
- Custom

The MasterTool IEC XE software offers a template called *MasterTool Standard Project*, which must be selected by the user as a model in the project creation. The new application will be developed according to a defined profile, also chosen by the user, adopting the predefined rules, features and patterns. Each project profile defines standard names for tasks and programs.

For compatibility guarantee of a project being developed using a specific profile, two approaches are used:

- MasterTool IEC XE allows the project creation only based on a template, selected at the same time as the profile to be used;
- At the code generation, MasterTool IEC XE checks all the defined rules for the user valid profile.

For details on Profiles see **Profiles**.

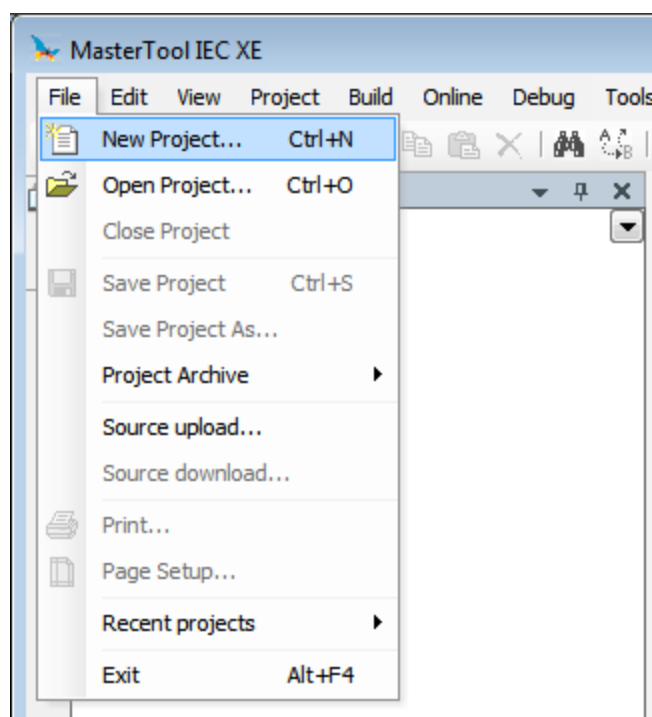
NOTE: While the project profiles are being described, some tasks are named, which are described in the **Task Configuration** section.

Start MasterTool IEC XE

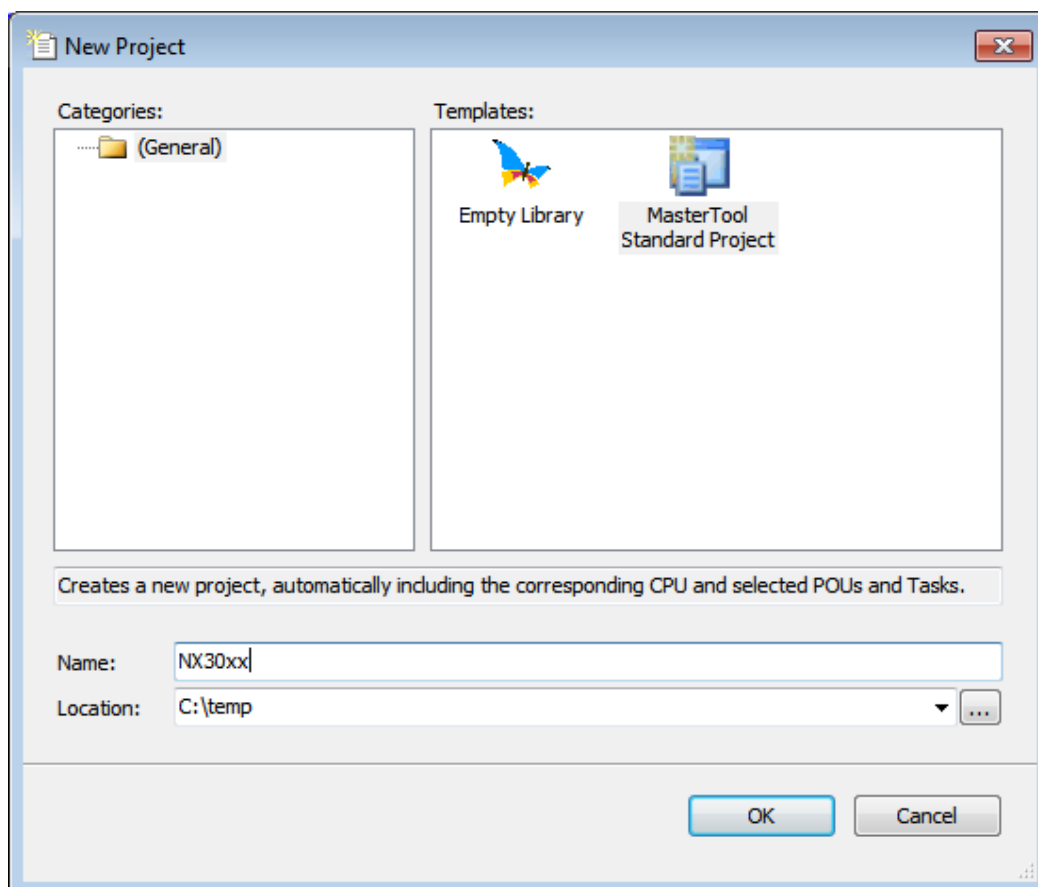
From the Start menu on your PC choose MasterTool IEC XE option.

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 4-1.

**Figure 4-1. 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 4-2. Project Classification**

Choosing the *MasterTool 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 CPU NX3010 (no redundancy) will be used, a NX9001 rack and a NX8000 power supply.

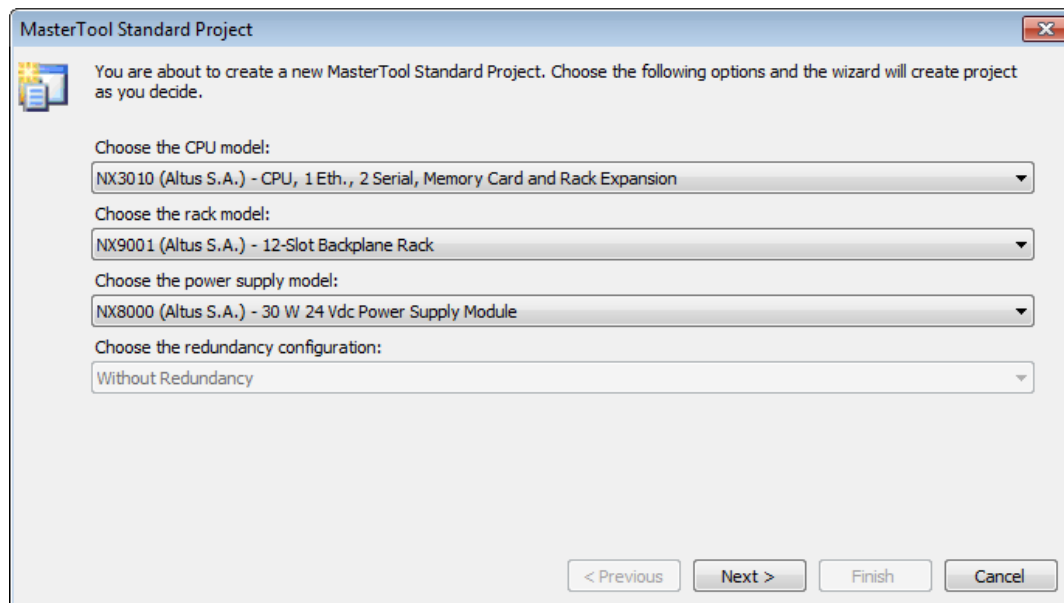


Figure 4-3. Hardware Modules

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

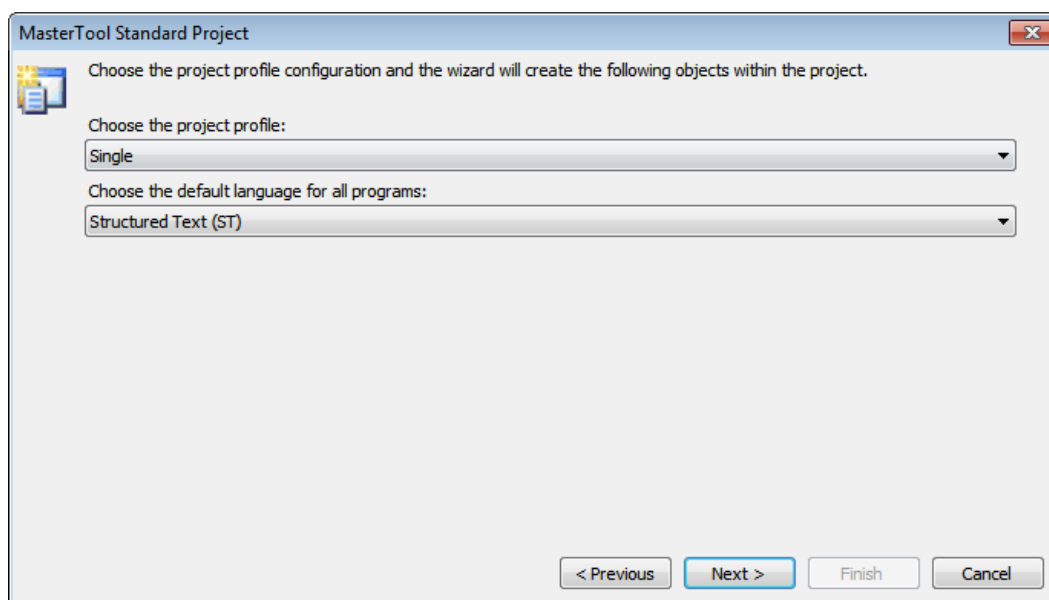


Figure 4-4. Profile Selection

The next screen defines the POU language created by the selected profile. As the profile is *Single*, there is only one POU (MainPrg) and the ST language remains. Click on *Previous* to return to the last screen, *Finish* to end or *Cancel* to cancel.

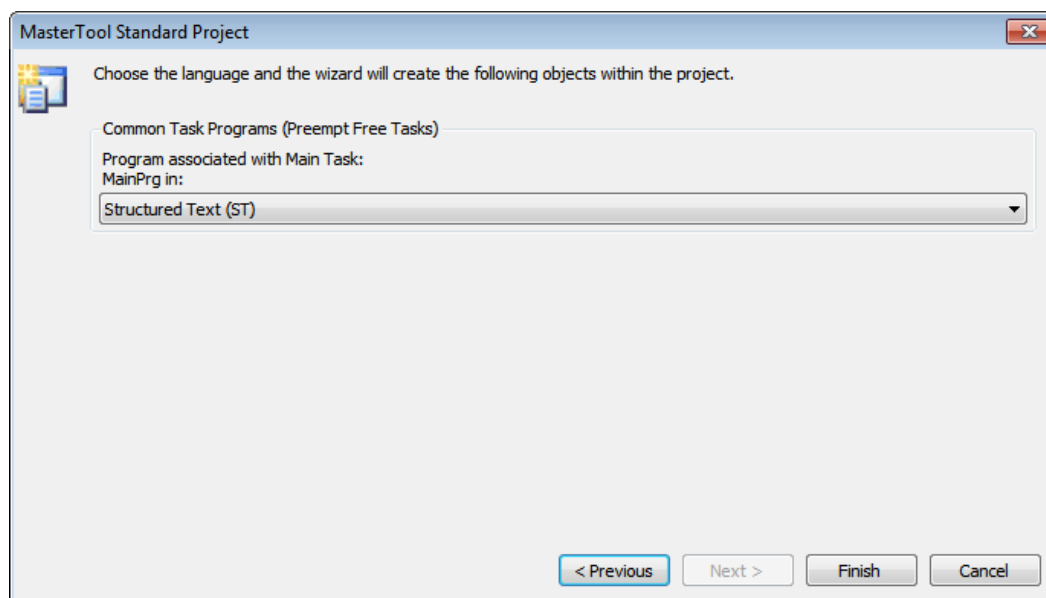


Figure 4-5. Programming Language

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

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 4-6:

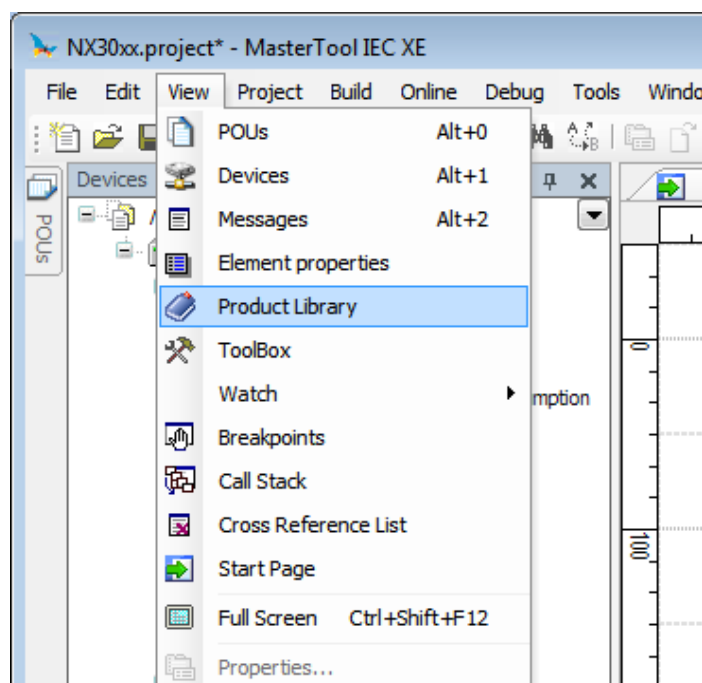


Figure 4-6. Library Visualization

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

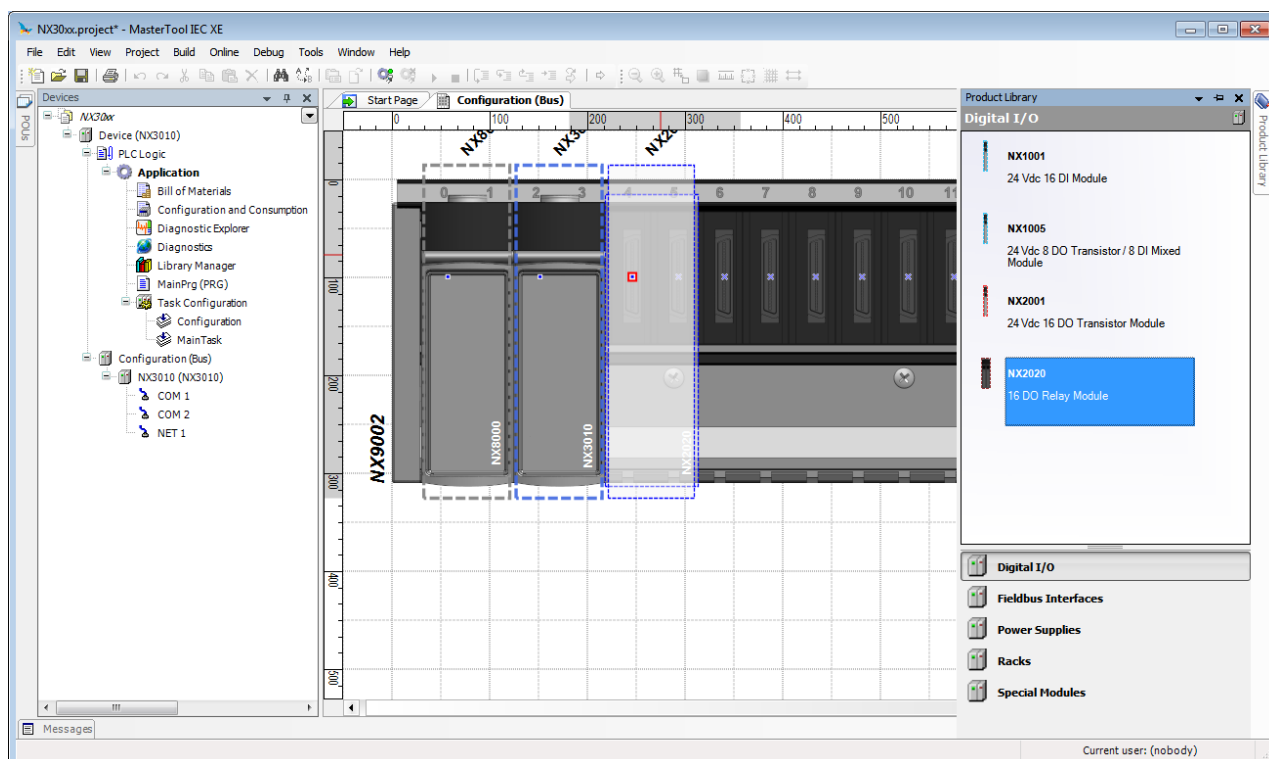


Figure 4-7. Adding Modules

Creating POU

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 POU are already created, but the user can create as many as he wants, limited by the program memory size.

To insert a new POU, one must click, using the mouse right button, on Application (default name created for the application), select *Add Object* and *POU...*, as depicted on Figure 4-8.

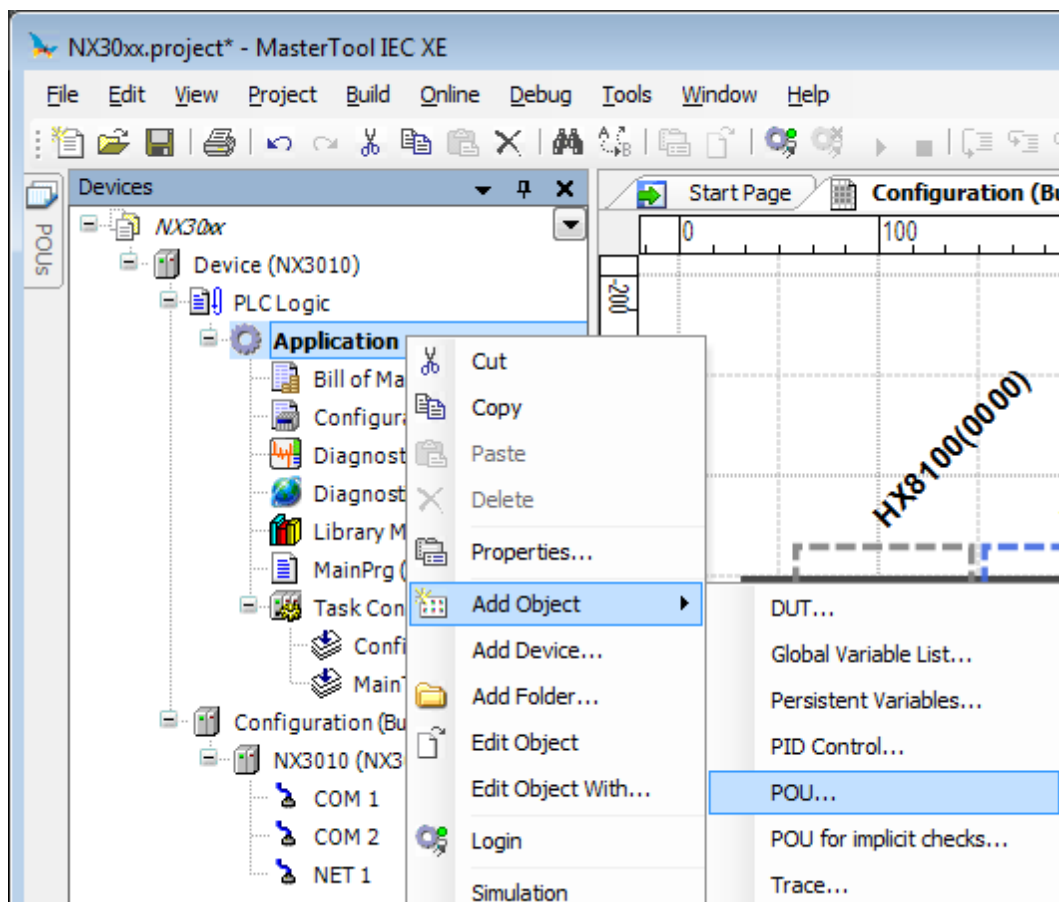
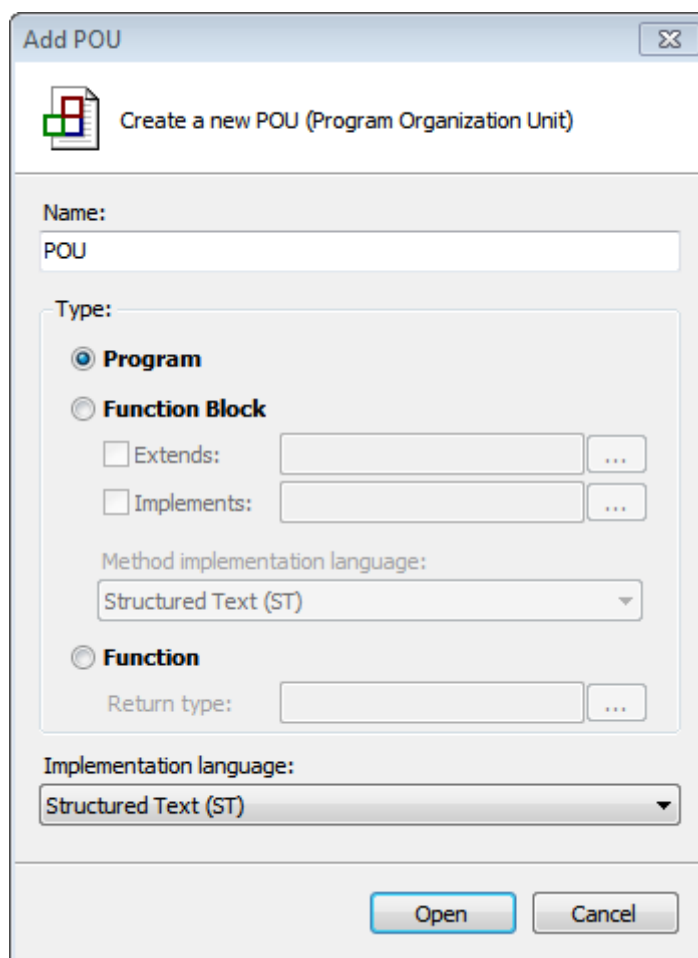


Figure 4-8. Inserting POU

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 *Open* must be clicked.



The image shows a software dialog box titled "Add POU" with a close button in the top right corner. Below the title bar is a header area with a document icon and the text "Create a new POU (Program Organization Unit)". The main area is divided into sections. The "Name:" section has a text field containing "POU". The "Type:" section contains three radio buttons: "Program" (selected), "Function Block", and "Function". Below "Function Block" are two checkboxes, "Extends:" and "Implements:", each followed by a text field and a three-dot menu button. Below these is a "Method implementation language:" dropdown menu set to "Structured Text (ST)". Below the "Function" radio button is a "Return type:" text field with a three-dot menu button. At the bottom of the main area is an "Implementation language:" dropdown menu also set to "Structured Text (ST)". At the very bottom of the dialog are "Open" and "Cancel" buttons.

Figure 4-9. Classification

For a POU editing, the tab with the correspondent name must be selected and the application development in the chosen language, started. The same procedure is valid for the POUs created automatically by the project profile.

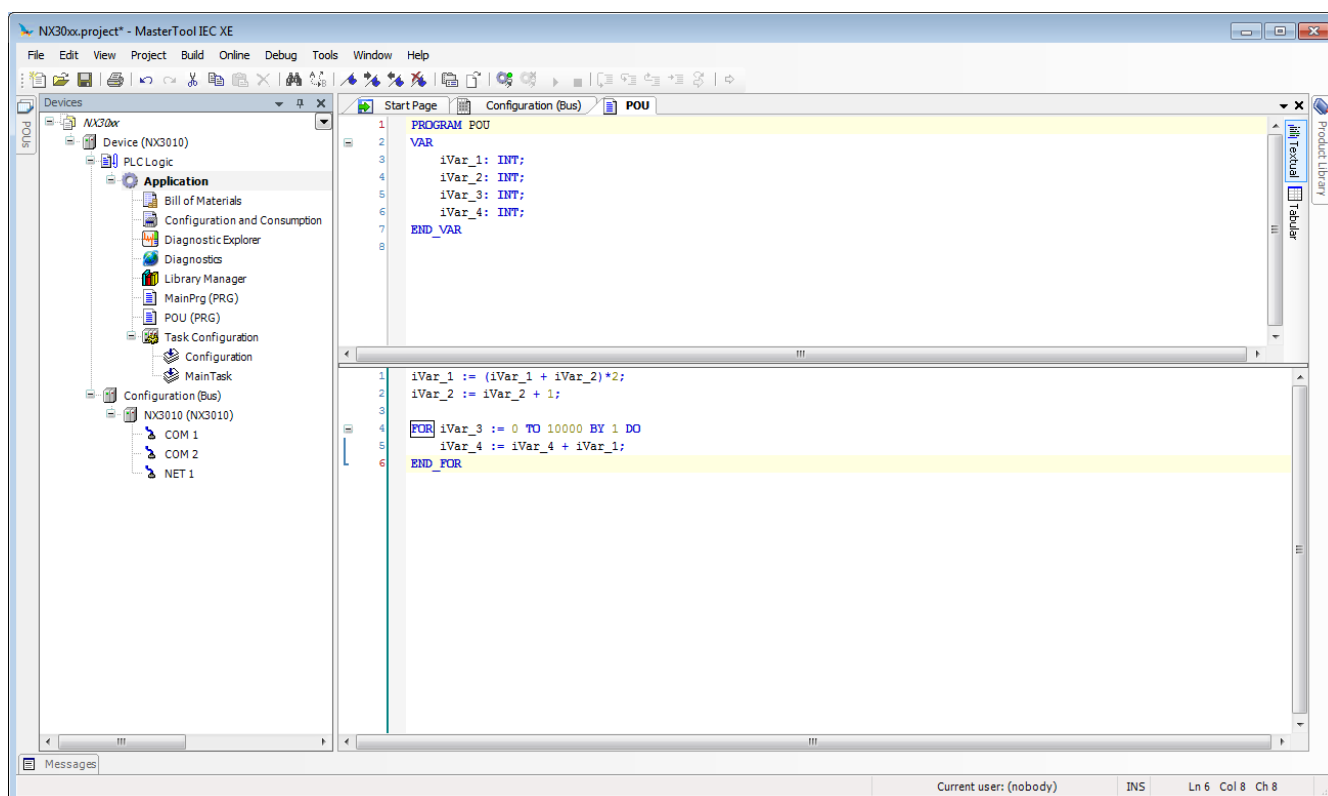


Figure 4-10. POU Editing

Creating Tasks

For a POU to be executed, it must be connected to a task. This scaling mechanism, called Task, is very useful for real time systems, which are defined by the periodic execution, or under request, of an event (change of state of any Boolean variable). The tasks control the program execution in different rates, according to the application features. The need to execute programs in different rates has the goal to reach the demand for process under control answer time and optimize the CPU processing capacity. The controllers that use 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 include a new task (in case the selected profile allows), the *Task Configuration* must be clicked using the right mouse button and the *Add Object* and *Task...* options must be selected, as shown on Figure 4-11.

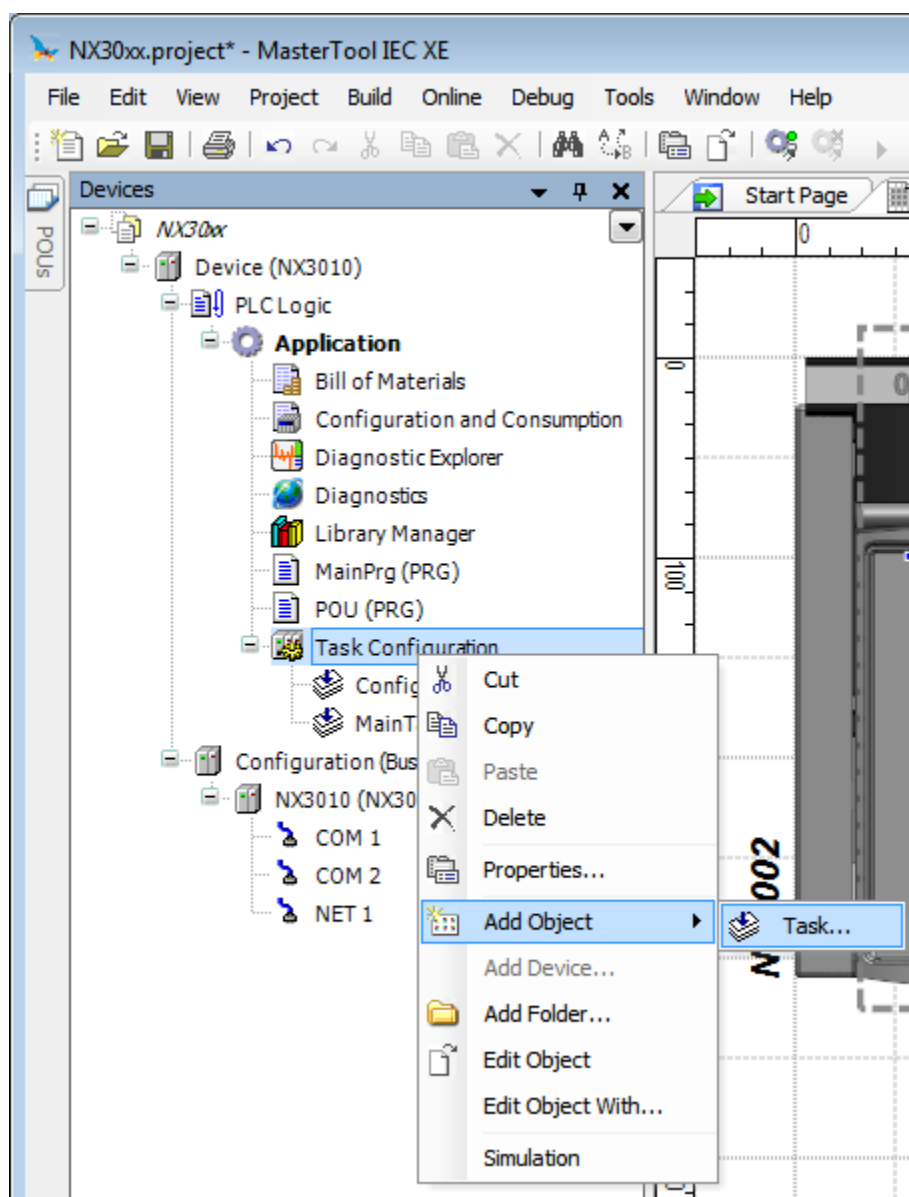


Figure 4-11. Creating a Task

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

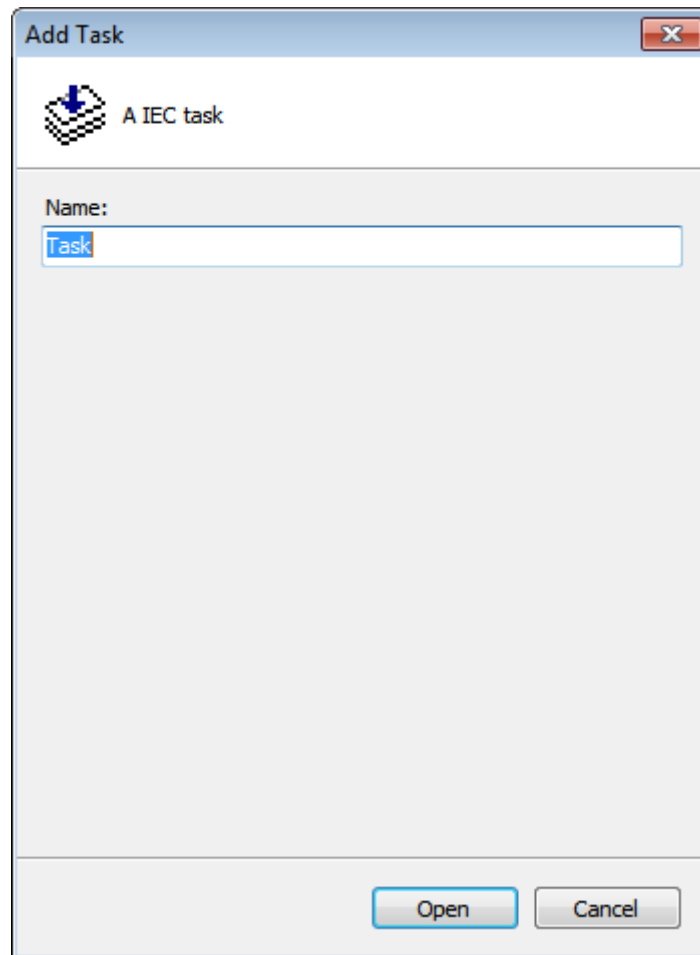


Figure 4-12. Task Name

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 define the priority, a number between 0 and 31, 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 with high priority for the system.

Type field define the type of the task, 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 from 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, the *Interval* must be configured (obligatory for cyclic type): the period of time, after which the task is called to run (maximum/minimum time for

MainTask is 750 ms and 5 ms, respectively. It is recommended to use a task cycle time (interval) of at least two times its execution time.

The CPU watchdog is configured to avoid the user tasks stopping.

The *Time* field sets the maximum time allowed for the execution of the task. If the task takes a longer time than watchdog to be executed, the application will to STOP and takes exception by watchdog.

The *Sensitivity* field indicates how many times the watchdog will have to be reached in order to confirm the exception. If the run time of the task reaches the Sensitivity field value multiplied by the Time field, the diagnosis will also be indicated. It should pay attention to the fact that the watchdog of the CPU is not used to protect the user's application of peaks at runtime but crashes. Therefore, your time should be configured with a high value, compared to the time of execution of the task to which it is related. The ideal is to maintain the average execution time of tasks in, at most, 50% of the time of watchdog. Thus, diminish the chances of watchdog errors for any peak time on task execution. Aiming to protect the system regarding to possible configuration error, the MasterTool IEC XE checks in all cyclic tasks, during the compilation, the watchdog (Software Watchdog) and the minimum and maximum limits of the task cycle time (Interval). It is important to highlight that the user will have to be careful when changing the pre-defined values by the project profiles as in this way it may put in risk the system execution. So, it is recommended to use the default values.

For further information see **Task Editor**.

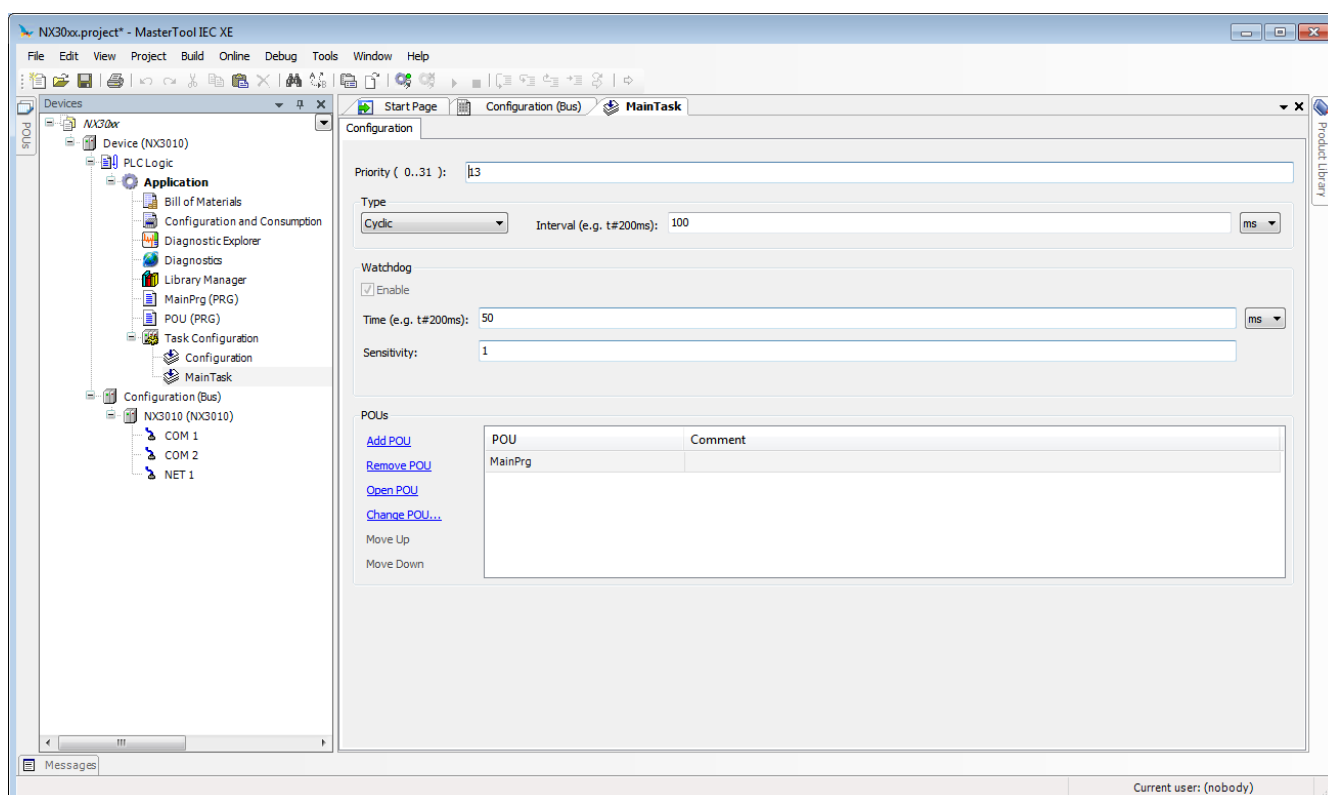


Figure 4-13. Configuring a Task

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* 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 4-14.

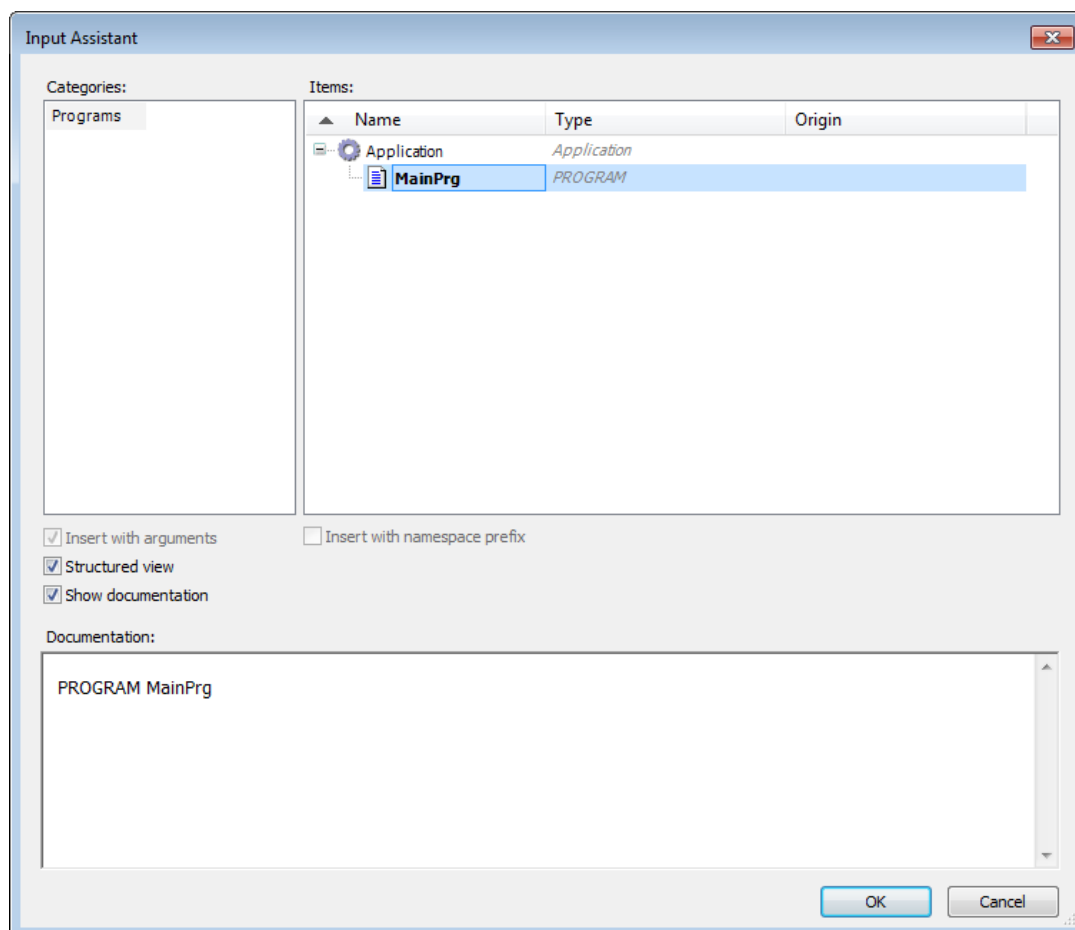


Figure 4-14. Connecting POUs to Tasks

Maximum Number of Tasks

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

For additional information on the maximum IEC task quantity per CPU and project profile, see User Manual Nexto Series CPUs NX3010, NX3020, and NX3030 (MU214605).

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 Nexto CPU, in the device tree and configure the field as described in the **Editors** chapter of this manual.

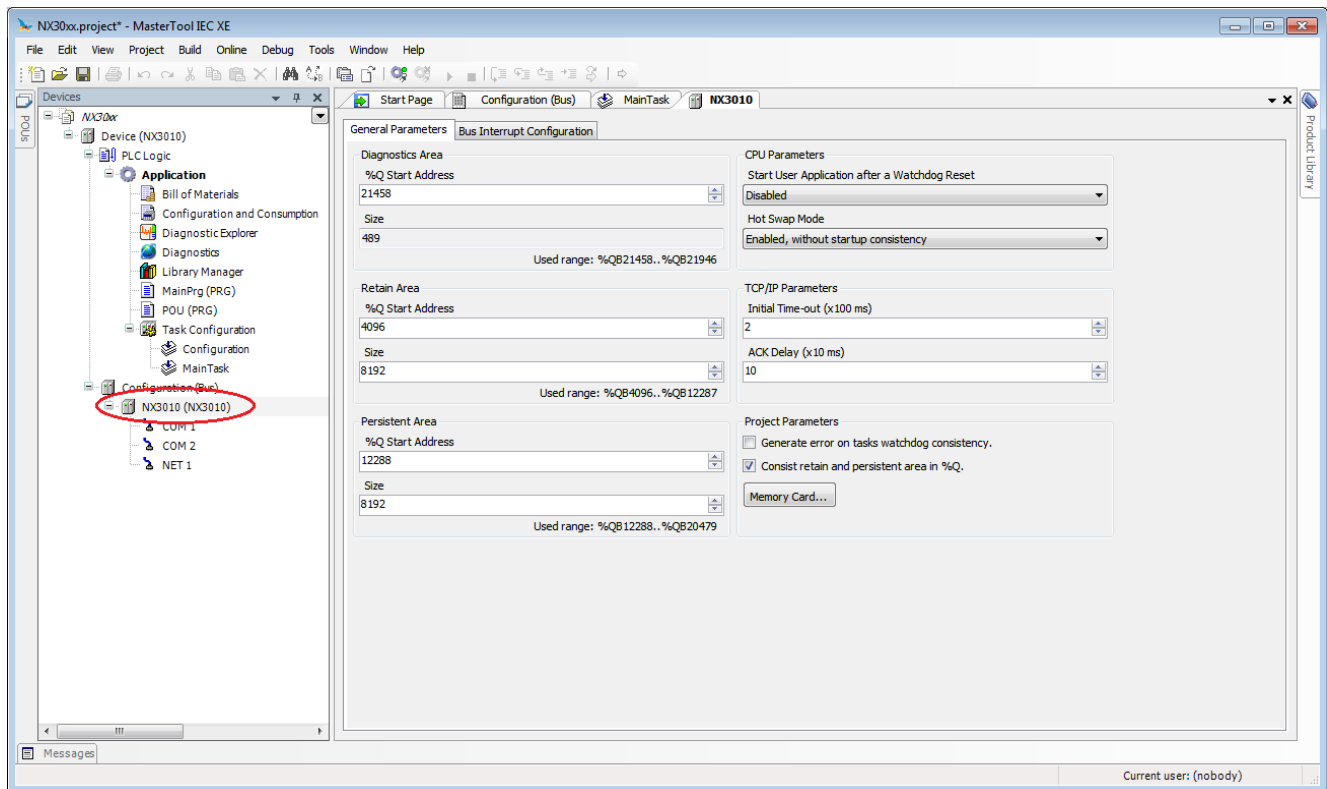


Figure 4-15. CPU Configuration

Besides that, for the communication between the CPU and the MasterTool IEC XE to be possible, the Ethernet NET 1 interface must be configured. Double-clicking on the CPU NET 1 icon, in the devices tree, a new tab will appear for the configuration of the communication network where the module is connected.

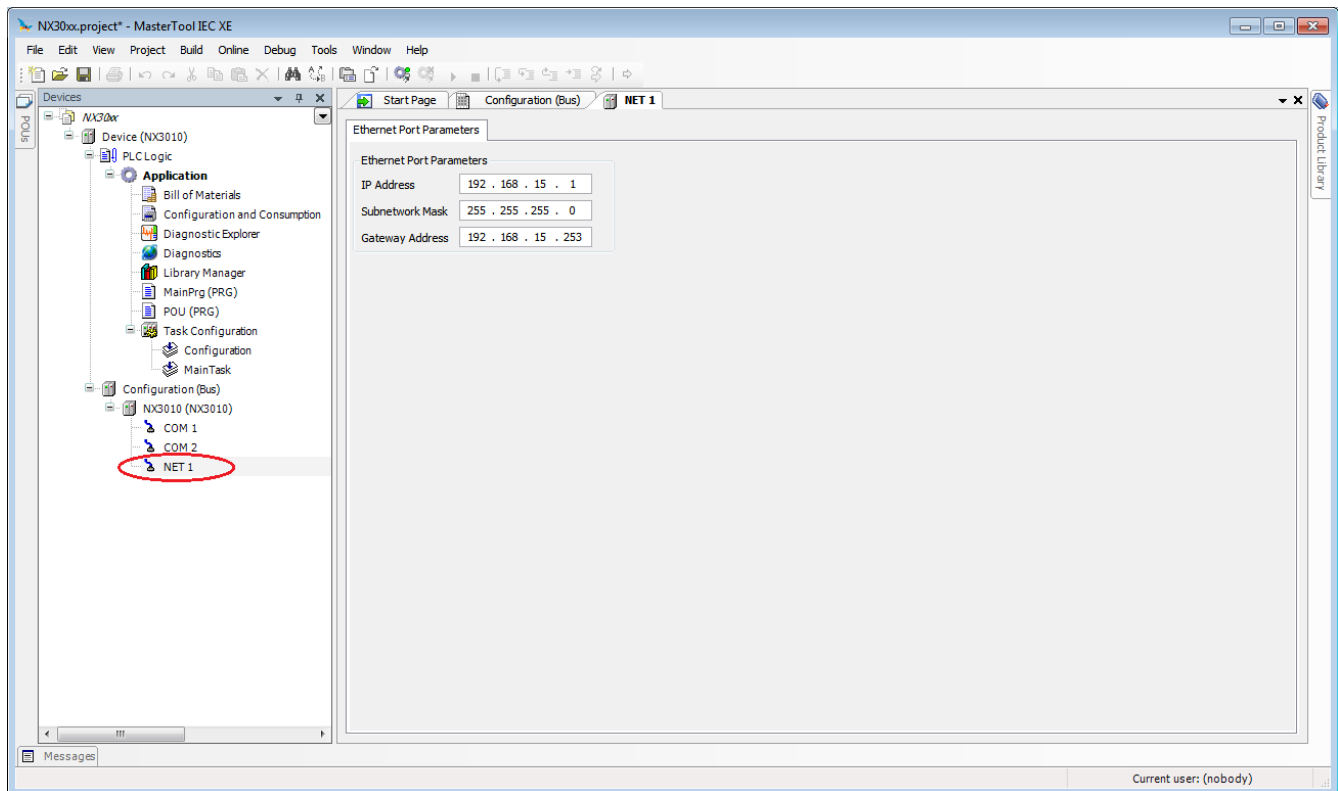


Figure 4-16. Configuring the CPU Communication Port

In case the CPU with the Ethernet configuration is not found in the network or the active CPU has a different Ethernet configuration (from the project), a message will appear on the screen during the *Login*, requesting to the user the possibility of changing the previous ETHERNET configuration by the configured (*Yes* option) or canceling and quit sending the project (*No* option).

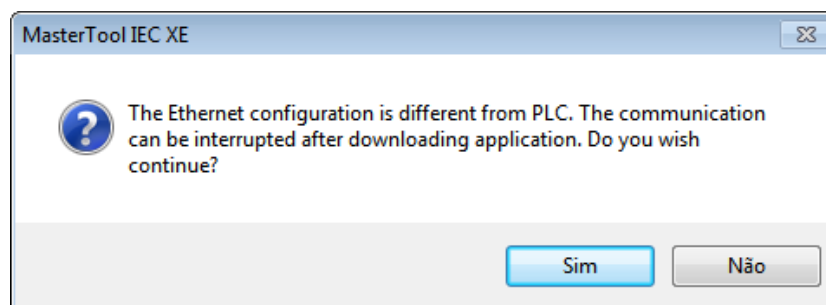


Figure 4-17. Configuration Warning

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 on Figure 4-18.

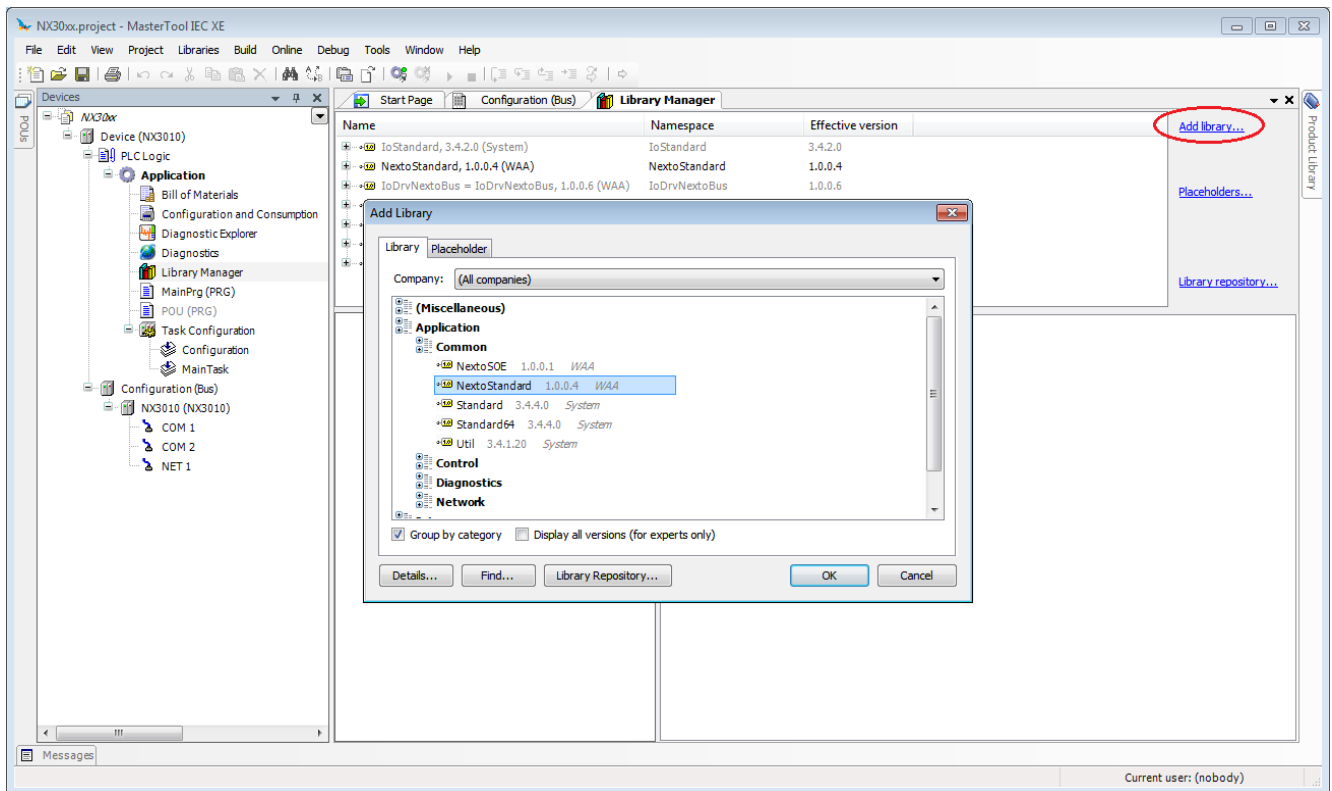


Figure 4-18. Inclusion of a Library in the Project

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. For further information see **Features**.

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.

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 on Figure 4-19.

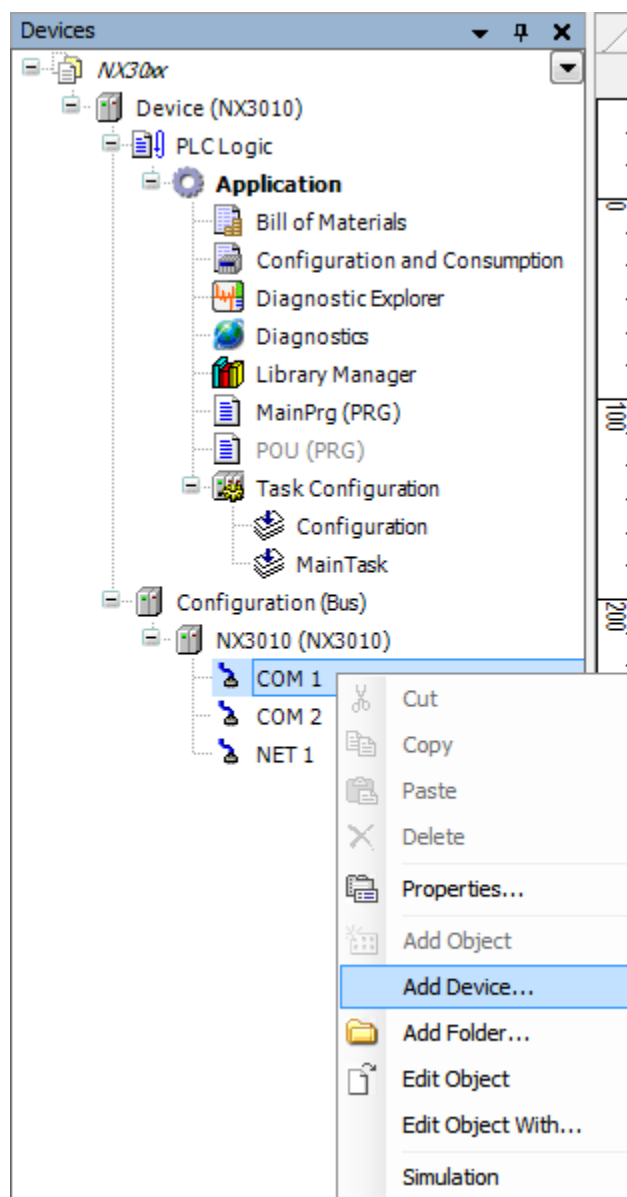


Figure 4-19. Adding an Instance

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*, as shown on Figure 4-20.

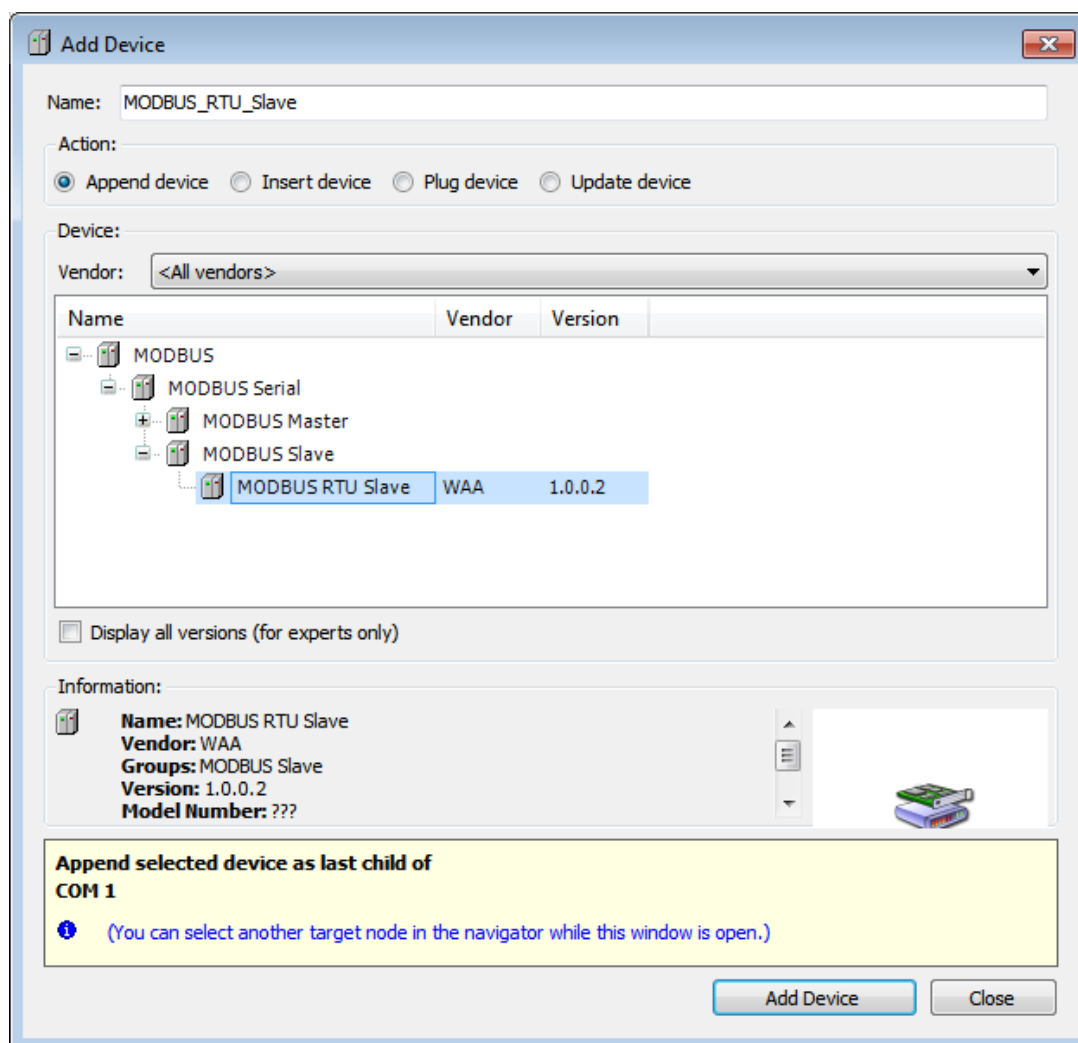


Figure 4-20. Selecting the Protocol

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, as the CPU NX3010 has only one Ethernet interface). Click on the NET with the mouse right button and select *Add Device...*, as shown on Figure 4-21.

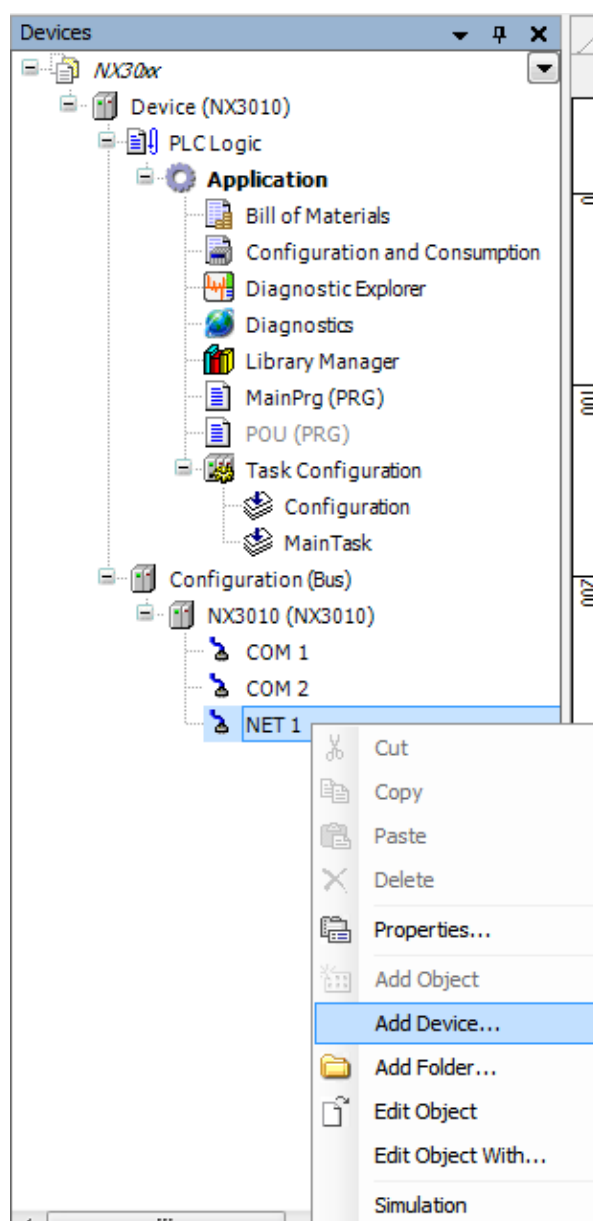


Figure 4-21. Adding an Instance

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*, as shown on Figure 4-22.

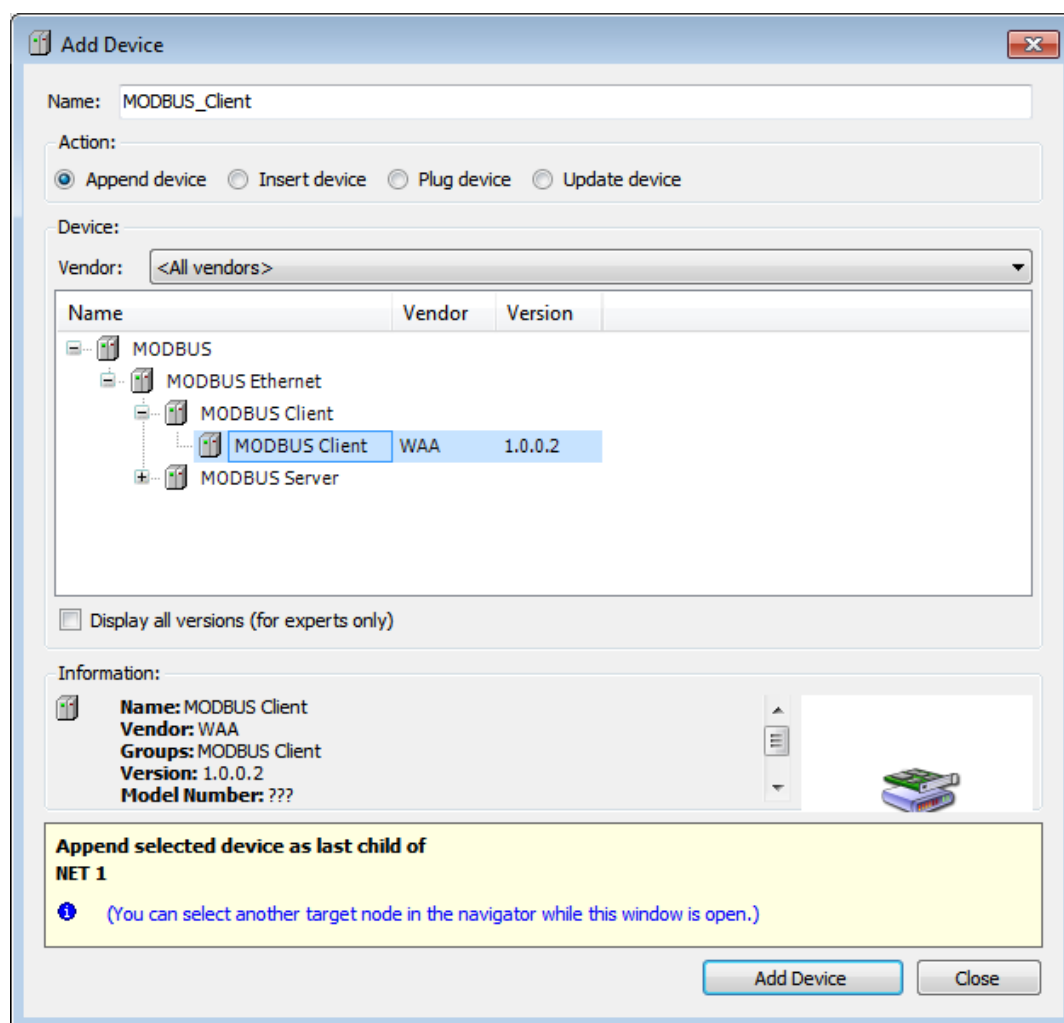


Figure 4-22. Selecting the Protocol

Finding the Network

As there is the possibility of more CPUs being connected to the network, the user must find all communication units and select the desired one.

Initially, the option Device must be accessed, in the device tree, double-clicking on it. In the tab *Communications Settings*, select the *Gateway* and click on *Scan network*.

Then, the user must wait until the MasterTool IEC XE software searches and show the available CPUs in the network.

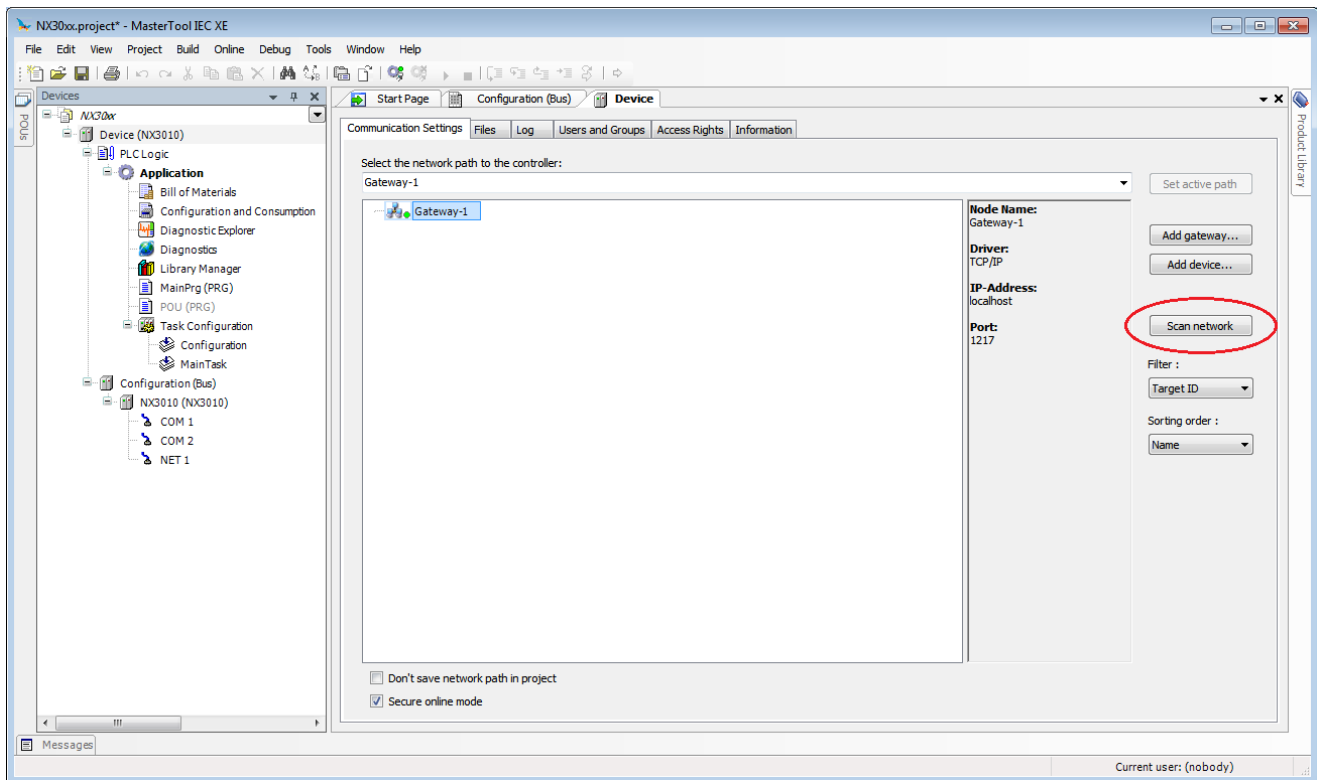


Figure 4-23. Finding the CPU

Following, the desired CPU must be selected and the option *Set active path* clicked, to activate the CPU and to inform the configuration software the CPU should communicate and send the project.

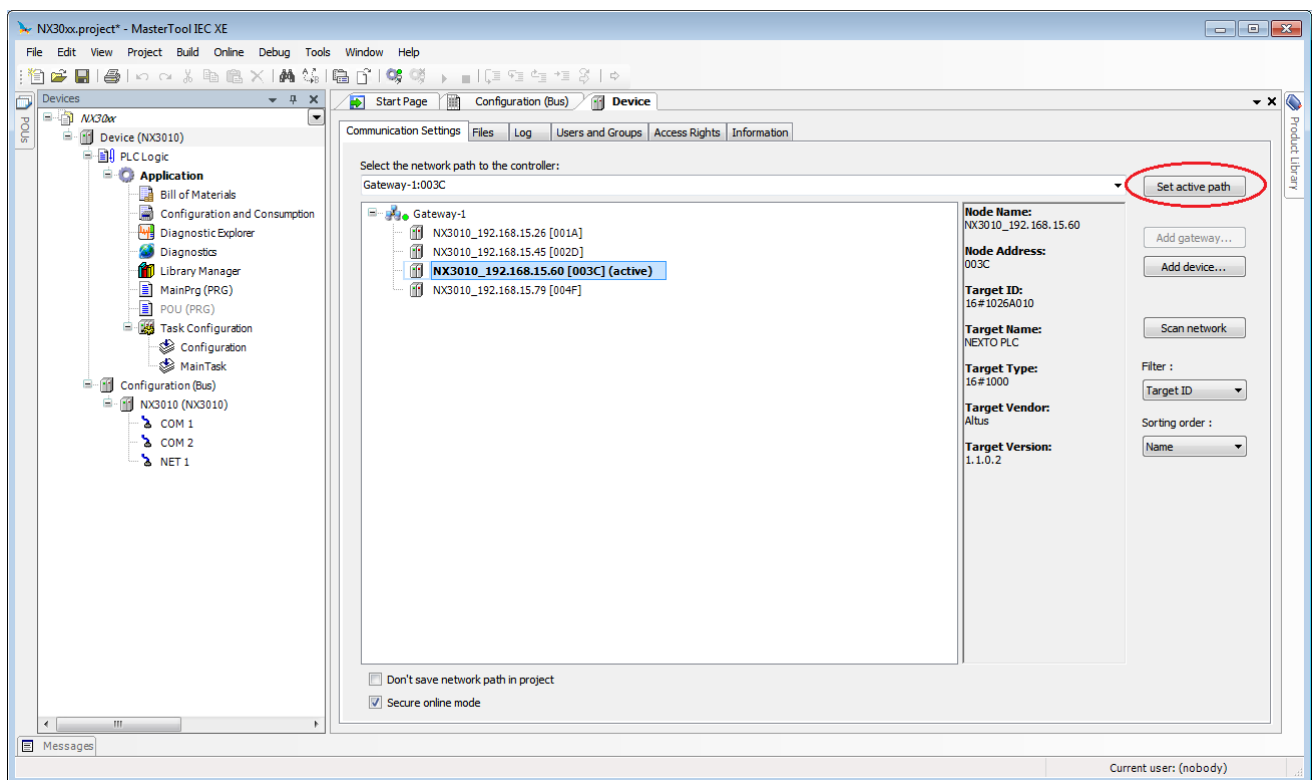


Figure 4-24. Activating the CPU

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*.

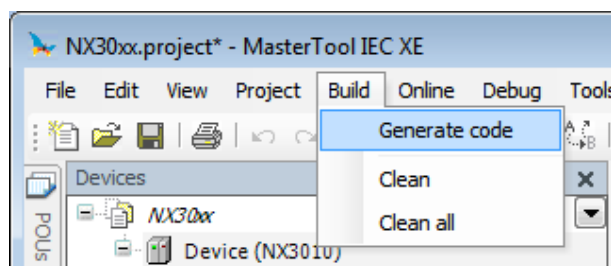


Figure 4-25. Building the Project

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 on Figure 4-26.

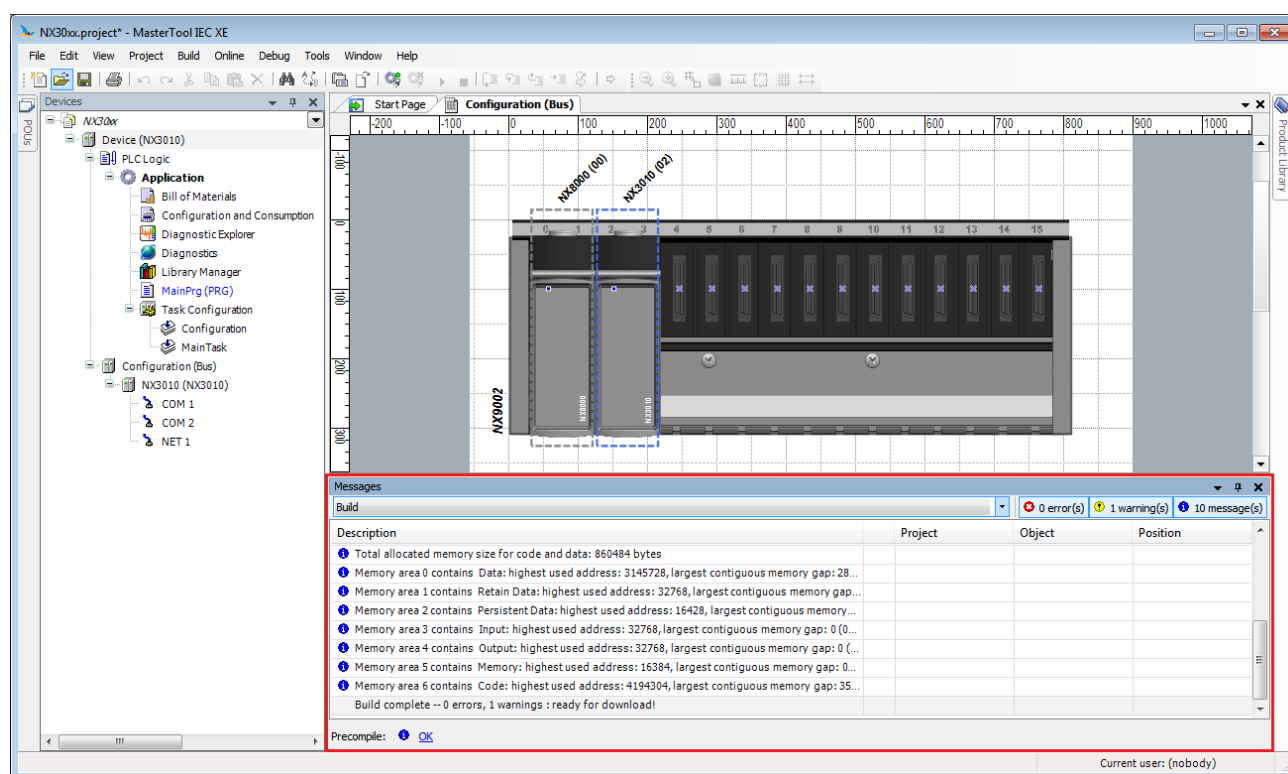


Figure 4-26. Building Messages

In case the errors and messages are not visible on the screen, the option *Messages* from the menu *View* must be selected, as shown on Figure 4-27.

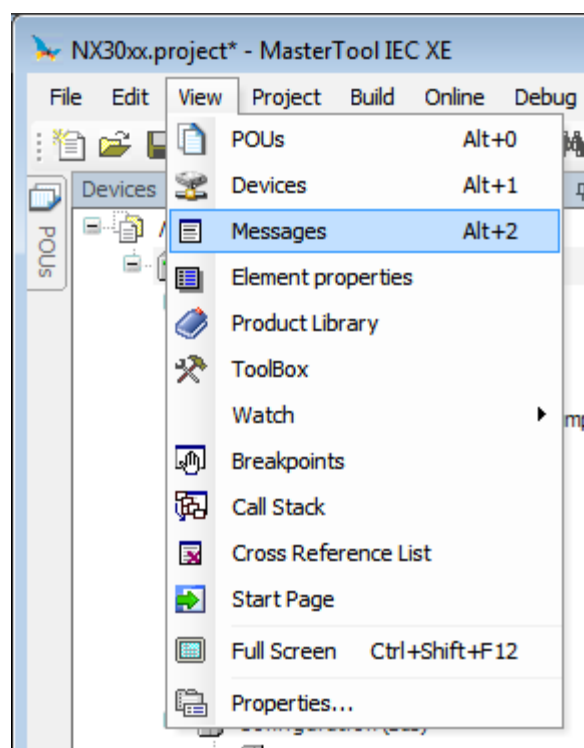


Figure 4-27. Including the Messages on the Screen

Login

After the application has been compiled and the errors found corrected, the project must be sent to the CPU. For that to be possible, the operation Login in the MasterTool IEC XE software must be executed. This operation may take a few seconds, depending on the size of the generated file. To execute the Login, go to *Online* menu and click on the option *Login*, as shown on Figure 4-28.

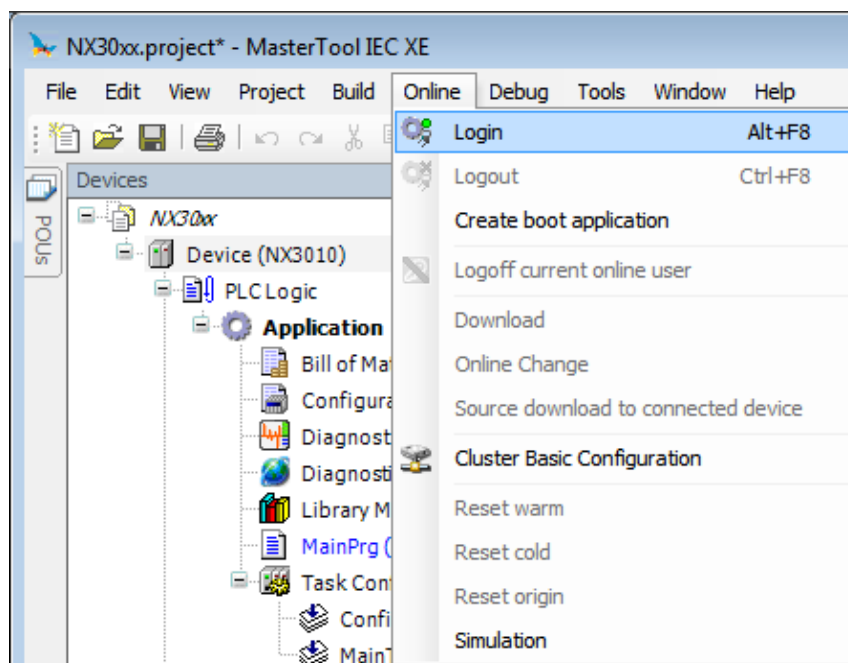


Figure 4-28. Sending the Project to the CPU

After the command execution, some user interface messages may appear, which are presented due to differences between an old project and the new project been sent, or simply because there was a variation in some variable.

Figure 4-29 shows the message the MasterTool IEC XE presents in case the new project, which is being sent, is different from the project already existent inside the CPU. The available options are the following:

- *Login with online change*: execute the login and send the new project without stopping the current CPU application (see **Run Mode**), updating the changes when a new cycle is executed.
- *Login with download*: execute the login and send the new project with the CPU stopped (see **Stop Mode**). When the application is initiated, the update will have been done already
- *Login without any change*: executes the login without sending the new project.

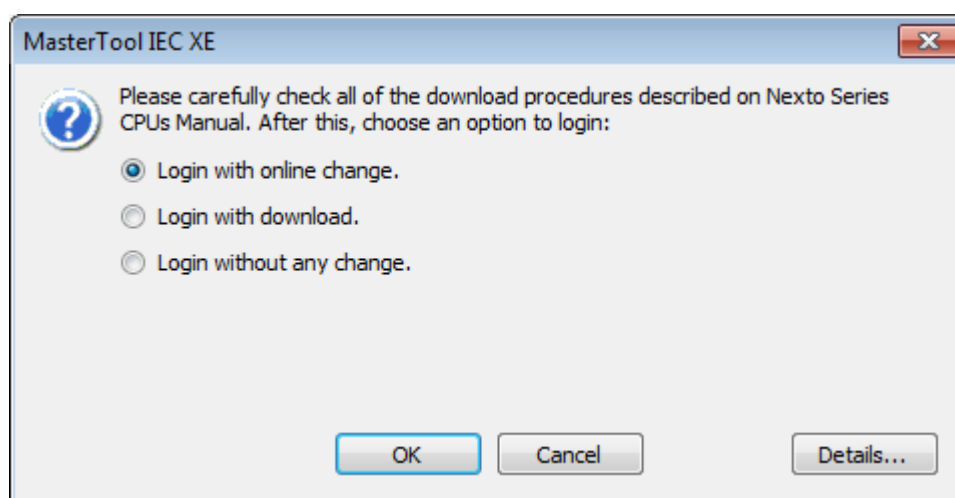


Figure 4-29. CPU Project Updating

Figure 4-30 shows a message the MasterTool IEC XE shows when only changes in the application variables were done; turning impossible the new project sending and updating on a CPU new cycle, which is in run mode (see **Run Mode Mode**). Therefore, the MasterTool IEC XE requests whether the login must be executed as download and the CPU stopped (see **Stop Mode Mode**) or the operation must be canceled.

The button *Details...* presents the changes made in the application.

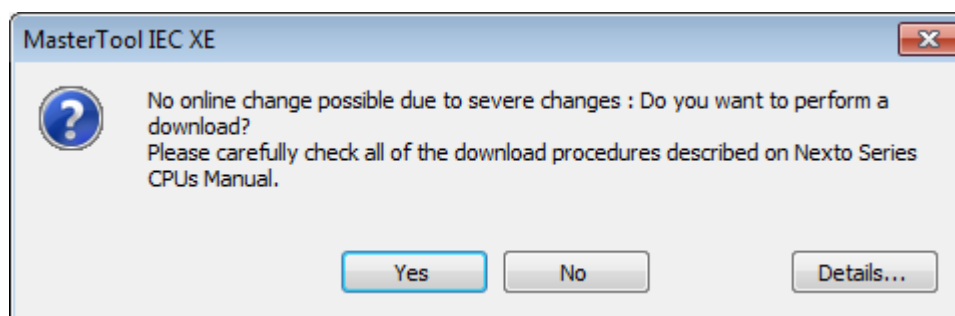


Figure 4-30. Variable Changes

At the first time, an application is sent to the CPU, the message shown on Figure 4-31 will appear on the MasterTool IEC XE screen.

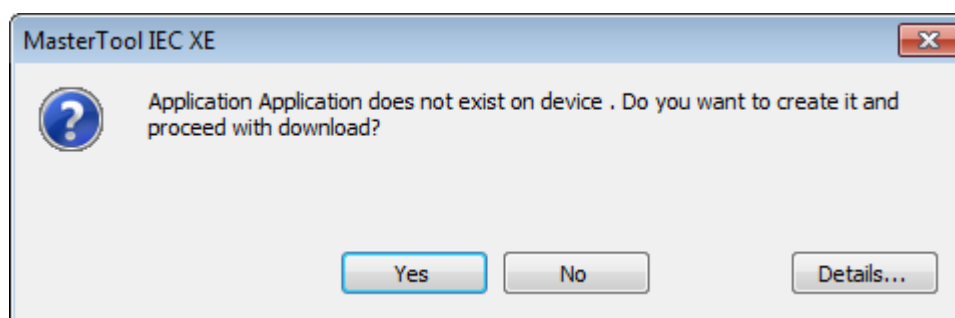


Figure 4-31. First Application Sending

Run Mode

Right after the project has been sent to the CPU, the application will not be immediately executed (only if an online – online change transmission were made). For that to happen, the command *Start* must be selected. This function allows the user to control the execution of the application sent to the CPU. Besides, it allows initial values to be pre-configured, in order to turn possible the CPU updating on the first cycle.

To select such functionality, the option *Start*, from the *Debug* menu, must be clicked, as shown on Figure 4-32.

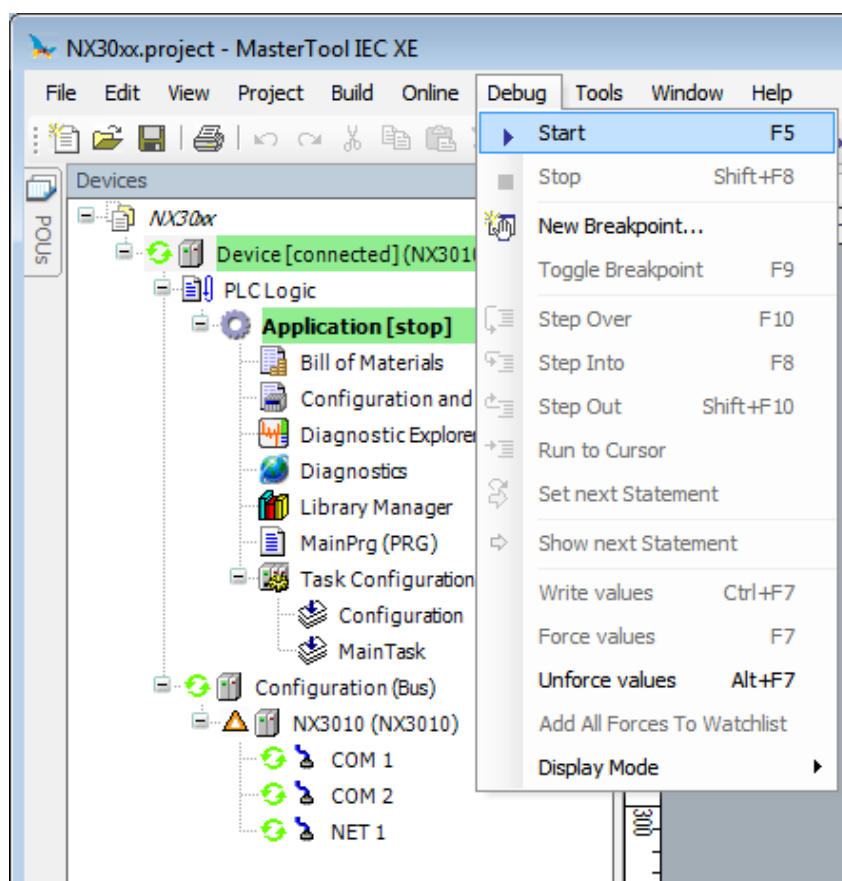


Figure 4-32. Starting the Application

Figure 4-33 shows the application in execution. In case the POU tab is selected, the created variables are listed on a monitoring window, in which the values can be forced and visualized by the user. In

case the variables are forced through the <F7> command on the keyboard, the CPU will indicate this condition on the graphic display.

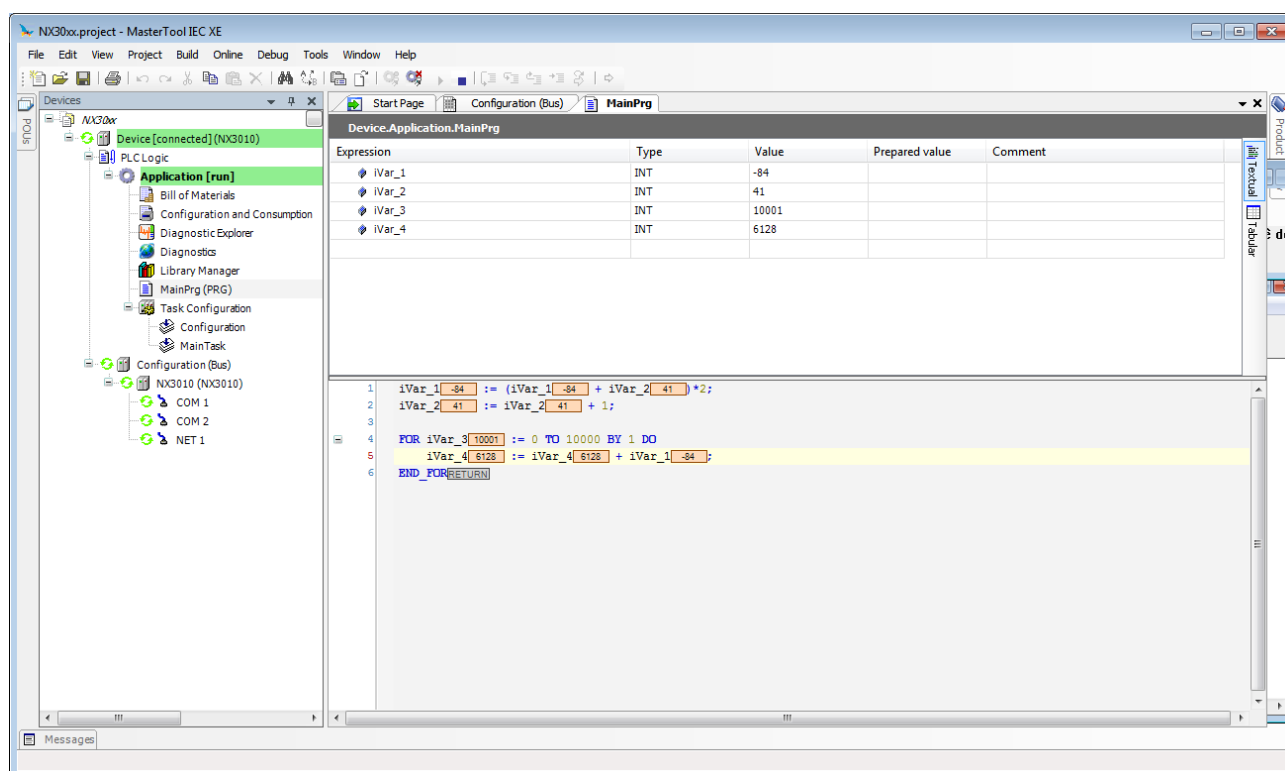


Figure 4-33. Program Being Executed

In case the CPU is initialized with an application already internally stored, it automatically goes to Run Mode, without the need for a MasterTool IEC XE command.

Stop Mode

For CPU execution interruption, without losing the connection with the MasterTool IEC XE software, the user must select the *Stop* option, available at the menu *Debug*, as shown on Figure 4-34.

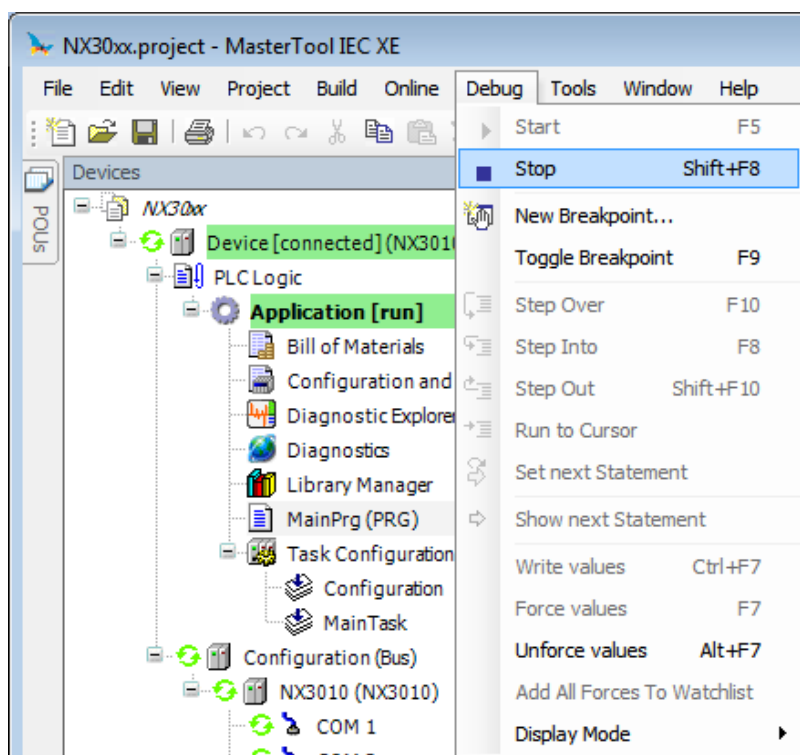


Figure 4-34. Stopping the Application

In case the CPU is initialized without the stored application, it automatically goes to Stop Mode, as it happens when a software exception occurs.

Logout

In case the user option is to finalize the communication with the CPU, the command *Logout* must be used, placed in the *Online* menu, as shown on Figure 4-35.

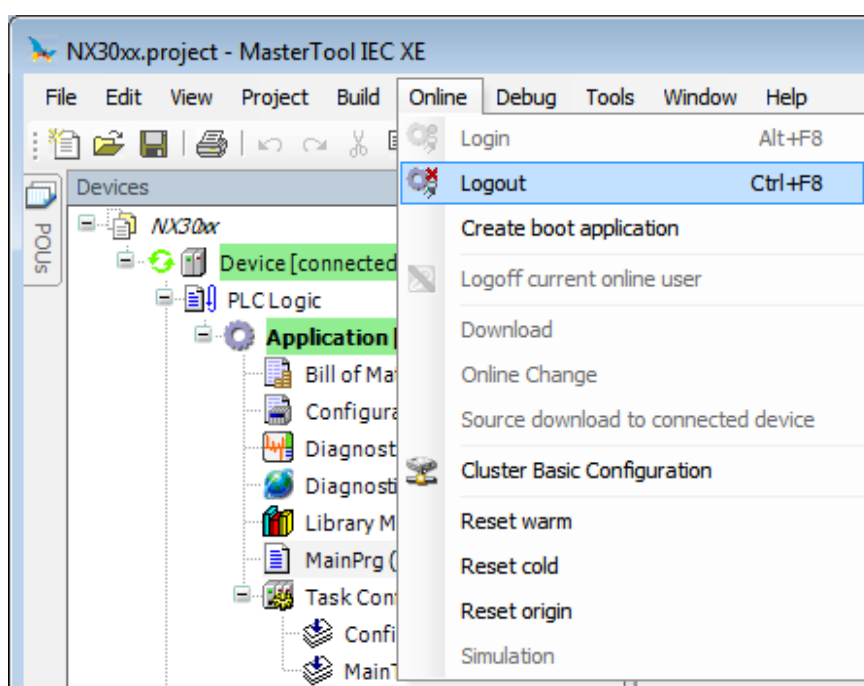


Figure 4-35. Interrupting the Communication with the CPU

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
- Diagnostics Explorer
- 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 Figure 4-36. After that a warning is displayed in the bottom bar of MasterTool IEC XE that indicates that the tool is operating in Simulation Mode.

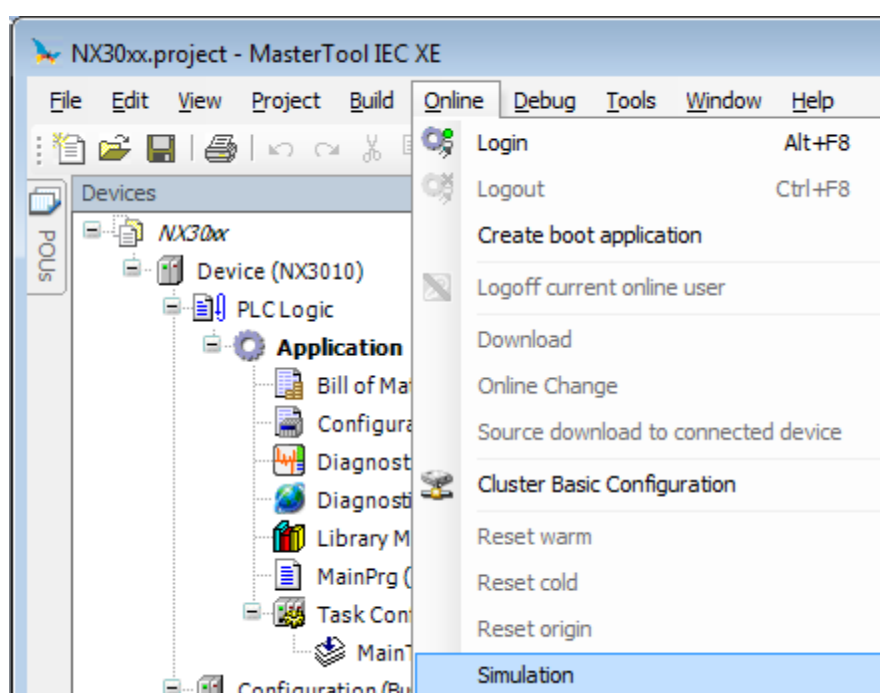


Figure 4-36. Simulation Mode

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 the 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.

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.

Project Upload

Nexto Series CPUs allow the project storing in the product memory, which can be uploaded and reused through the MasterTool IEC XE software.

To store a project in the product memory, the CPU must be connected (*Login*) and the project sending option must be selected, together with the applicative.

To upload the project previously stored, the options shown on Figure 4-37 must be selected.

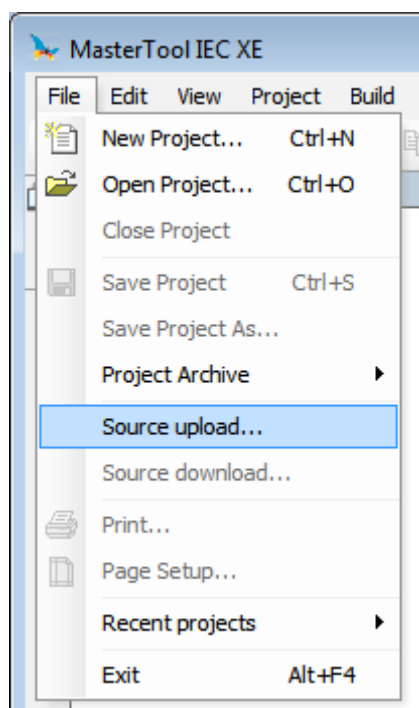


Figure 4-37. Project Upload Option

After, the desired CPU has to be selected and the *OK* button clicked as shown on Figure 4-38.

To ensure that the project of the CPU is completely equal and can be loaded without problems from other stations, see chapter **Projects Download/Login Method without Project Differences**.

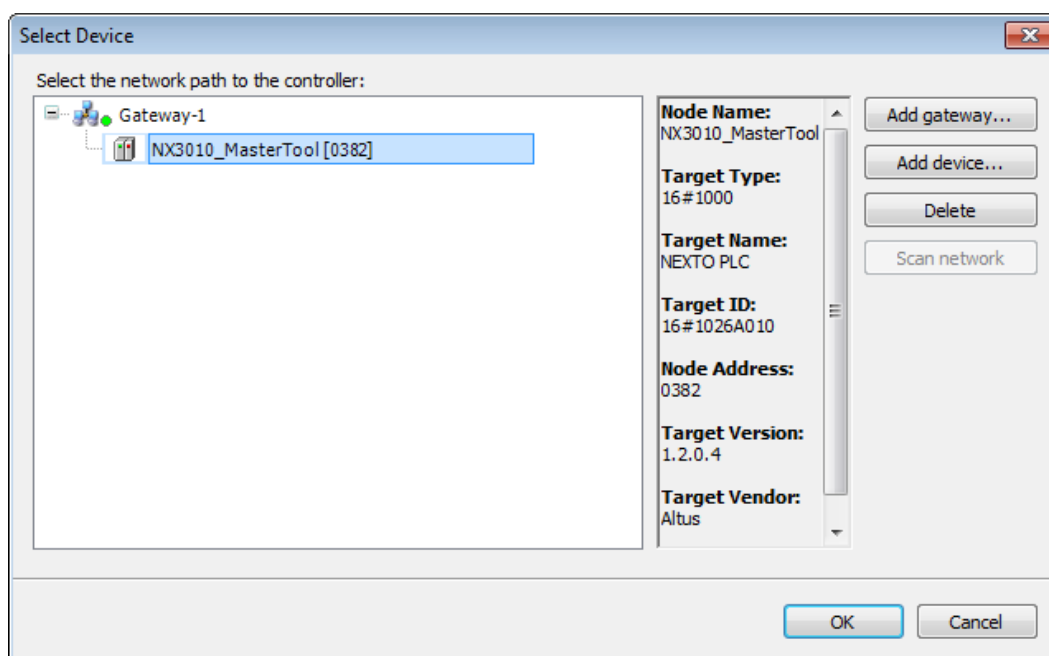


Figure 4-38. Select the CPU

ATTENTION:

The memory size area to store a project in the Nexto CPUs is defined on each CPU model documentation.

ATTENTION:

The upload recovers the last project stored in the controller as described in the previous paragraphs. In case only the download for execution of a specific applicative occurs, it will not be possible to be recovered through Upload.

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 (MainPrg) and a function block (FB1). MainPrg will contain a counter variable (ivar) and call function block (FB1). FB1 will get input “in” from MainPrg, will add “2” on this input and will write the result to an output “out”. “Out” will be read by MainPrg.

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

Declare Variables in MainPrg

The POU “MainPrg”, which by default is already available in the *Devices* window, is automatically opened in a ST language editor window in the center part of the MasterTool IEC XE user interface.

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 MainPrg) and the embracing keywords “VAR” and “END_VAR” for the variables declaration.

The body is empty, only line number 1 is displayed (Figure 4-39).

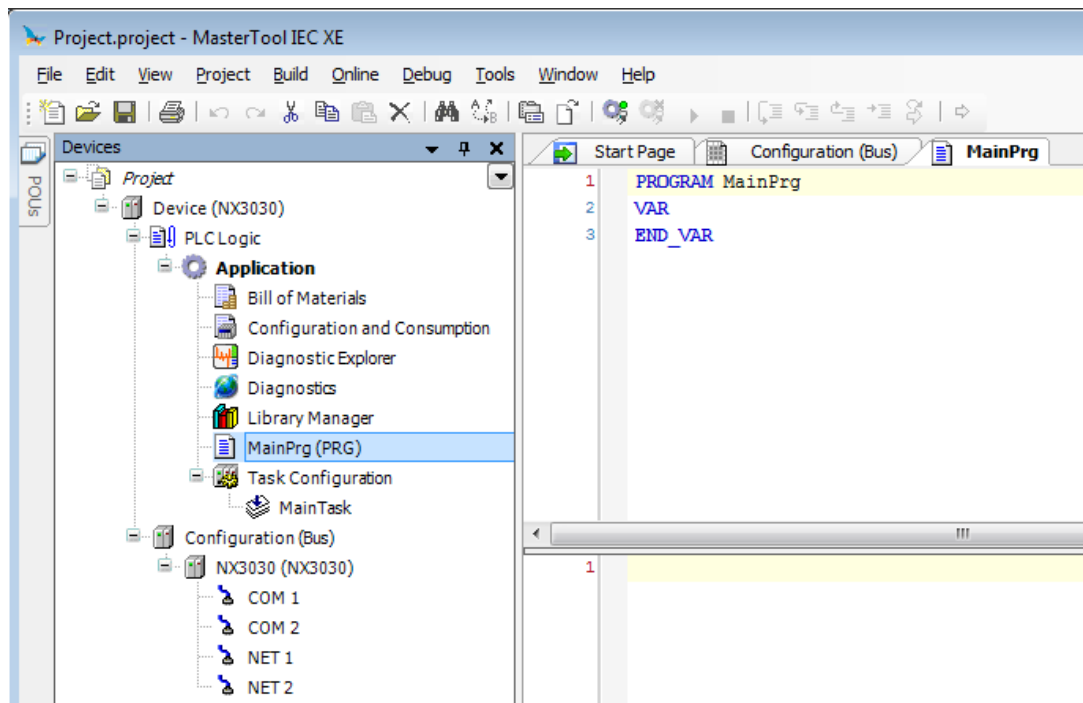


Figure 4-39. ST Editor Window

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 MainPrg
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.

Enter Programming Code in the Body of MainPrg

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

Instead of steps you can use the Auto Declaration feature: Without an preceding declaration enter an instruction immediately in the body of the program, then press the <ENTER> key. For each not yet declared variable found in the implementation line the *Auto Declare* dialog will open, where you now can carry out the declaration settings.

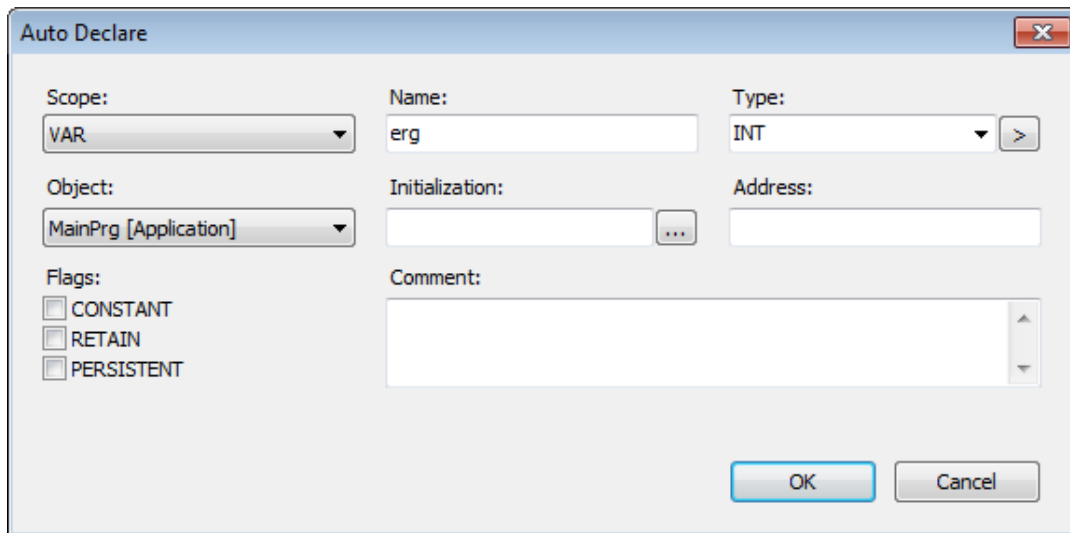


Figure 4-40. Auto Declare Dialog

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 (Figure 4-41).

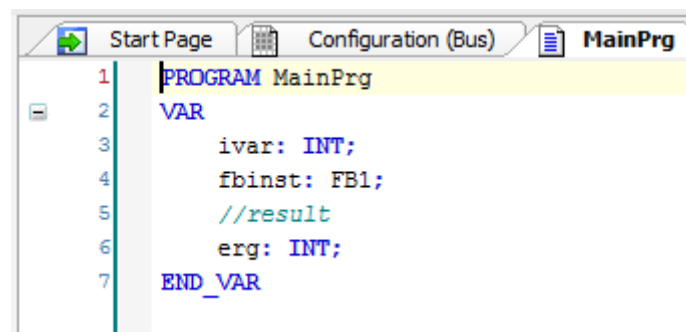


Figure 4-41. “erg” Variable Declaration

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 *Open* 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 MainPrg the following variables:

```
FUNCTION_BLOCK FB1
VAR_INPUT
    in:INT;
```



```

END_VAR
VAR_OUTPUT
    out:INT;
END_VAR
VAR
    ivar:INT:=2;
END_VAR

```

In the implementation part of the editor enter the following:

```
out:=in+ivar;
```

Define the Resources for Running and Controlling the Program on Nexto

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 SysTray*, 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.

Start Nexto (simulation)

The simulation device PLC (MasterTool IEC XE SP Win) is available as a service at system start. It is represented by an icon in the system tray: () for status “stopped”, () for status “running”. If allowed by the system, this device will be automatically started at system start. Otherwise, you have to start it manually by command *Start PLC* from the menu you open by a mouse-click on the icon.

This icon is part of the MasterTool IEC XE. It provides a menu with a *Start* and a *Stop* commands, thus allowing the user to stop or restart the service manually. The menu also includes the command *Exit PLC Control*, which just terminates the GatewaySysTray program, not however the PLCservice.

Configure a Communication Channel to Nexto

The resulting connection will finally be entered in the line below Select network path to the *Nexto controller*.

Perform a double-click on entry *Device* in the *Devices* window. The dialog *Device* will open with subdialog *Communication Settings*. Here you have to set up the connection between Nexto and the programming system according to the item *Finding the Network*.

Run and Watch the Application on Nexto

Compile and Load Application on Nexto

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 Nexto. Therefore make sure, that Nexto 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.

Starting the Application on Nexto

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.

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 POU

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 *MainPrg* 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 Nexto.

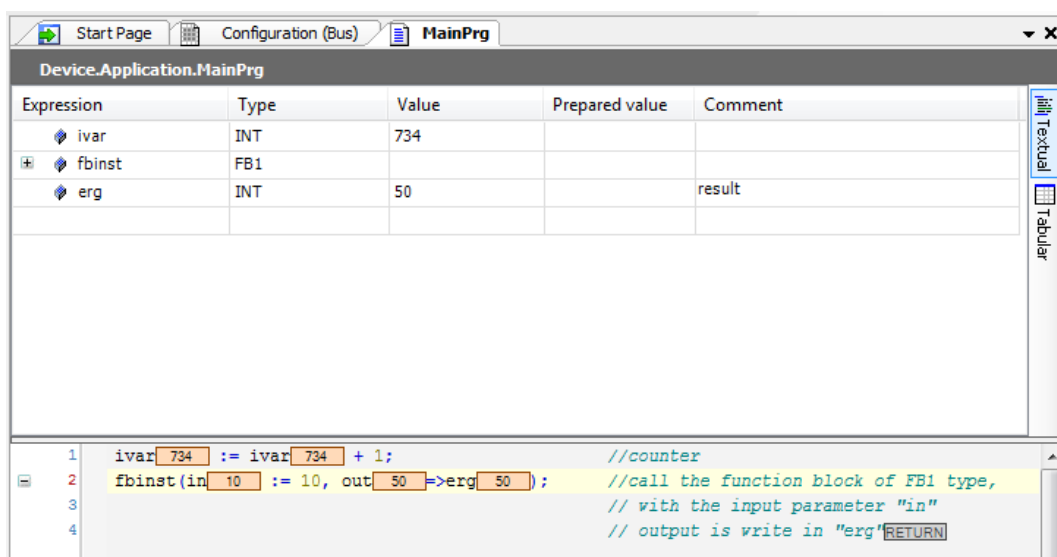


Figure 4-42. Monitoring Expressions and Code Lines

Writing and Forcing Variables

You can write or force a *Prepared value* to variable “ivar” on the Nexto, which means that “ivar” will be set to this value at the beginning of the next cycle. Perform a double-click on the field in column *Prepared value*, enter a desired integer value and leave the field by <ENTER> or by a mouse-click outside of the field. Perform command *Write Values* and *Force Values* (*Debug* menu) to write or force this value on Nexto. You will see the result at once in column *Value*.

Using the Watch Views

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

From the *View* menu use *Watch*, *Watch 1* commands. The watch window will open.

In column *Expression* perform a mouse-click in the first line of the table to open an edit frame. Enter the complete path for variable “ivar” which should be monitored: “Device.Application.MainPrg.ivar”.

It is recommended to use the input assistant via button for this purpose. Close the edit frame with <ENTER>. The type will be added automatically.

Do the same for the further variables. The watch list shown in the next picture just contains expressions of MainPrg, but of course, the user might create a set of any variables of your project. Notice that for instance variables, for example for the FB1-instances, it is sufficient to enter the expression “Device.Application.MainPrg.fbinst”. The particular variables will be entered automatically and the corresponding lines can be opened via the “+” symbol: The current value of a variable is displayed in the *Value* column:

Expression	Type	Value	Prepared value	Comment
Device.Application.MainPrg.ivar	INT	6798		
Device.Application.MainPrg.fbinst	FB1			
Device.Application.MainPrg.erg	INT	50		result

Figure 4-43. 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 Nexto and the current value will be displayed in column *Value*:

Expression	Type	Value	Prepared value	Comment
Device.Application.MainPrg.ivar	INT	7403		
Device.Application.MainPrg.fbinst	FB1			
in	INT	10		
out	INT	62		
Device.Application.MainPrg.erg	INT	62		result

Figure 4-44. Current Value

Writing and forcing values is also possible.

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

Debug an Application

Set Breakpoint and Step Through the Program

In online mode you can set breakpoints as defined halt positions for the program execution.

When the program has reached a breakpoint you can execute the program in single steps. At each halt position you see the current value of the variables in the monitoring views.

Select line 1 of MainPrg. Press key <F9>, which equals the command *Toggle Breakpoint* from the *Debug* menu. The breakpoint will be indicated.

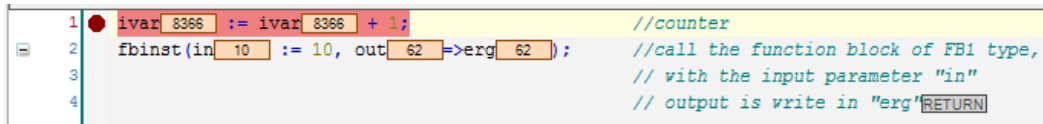


Figure 4-45. Application in “Stop” State

A running application will stop at a breakpoint:

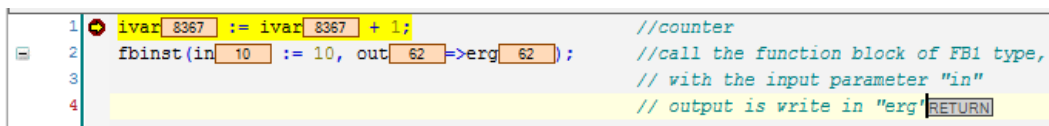


Figure 4-46. Application in “Run” State

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

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

You might also have a look at the breakpoints dialog to be opened via command *Breakpoints* from the *View* menu. Here, the breakpoints currently set can be viewed and edited and new breakpoints might be entered.

Notice also that the breakpoint positions will be remembered when you log out. They will be indicated by faded red bullets.

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.

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.

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

User Interface Components

The MasterTool IEC XE programming user interface is an arrangement of “components” (see below Figure 5-1).

This interface depends on the arrangement of windows, which can be modified by the user anytime via shifting, docking/undocking views, resizing or closing windows.

The user interface provides menus and toolbars, editor and object organization and watch and message windows and an information and status line.

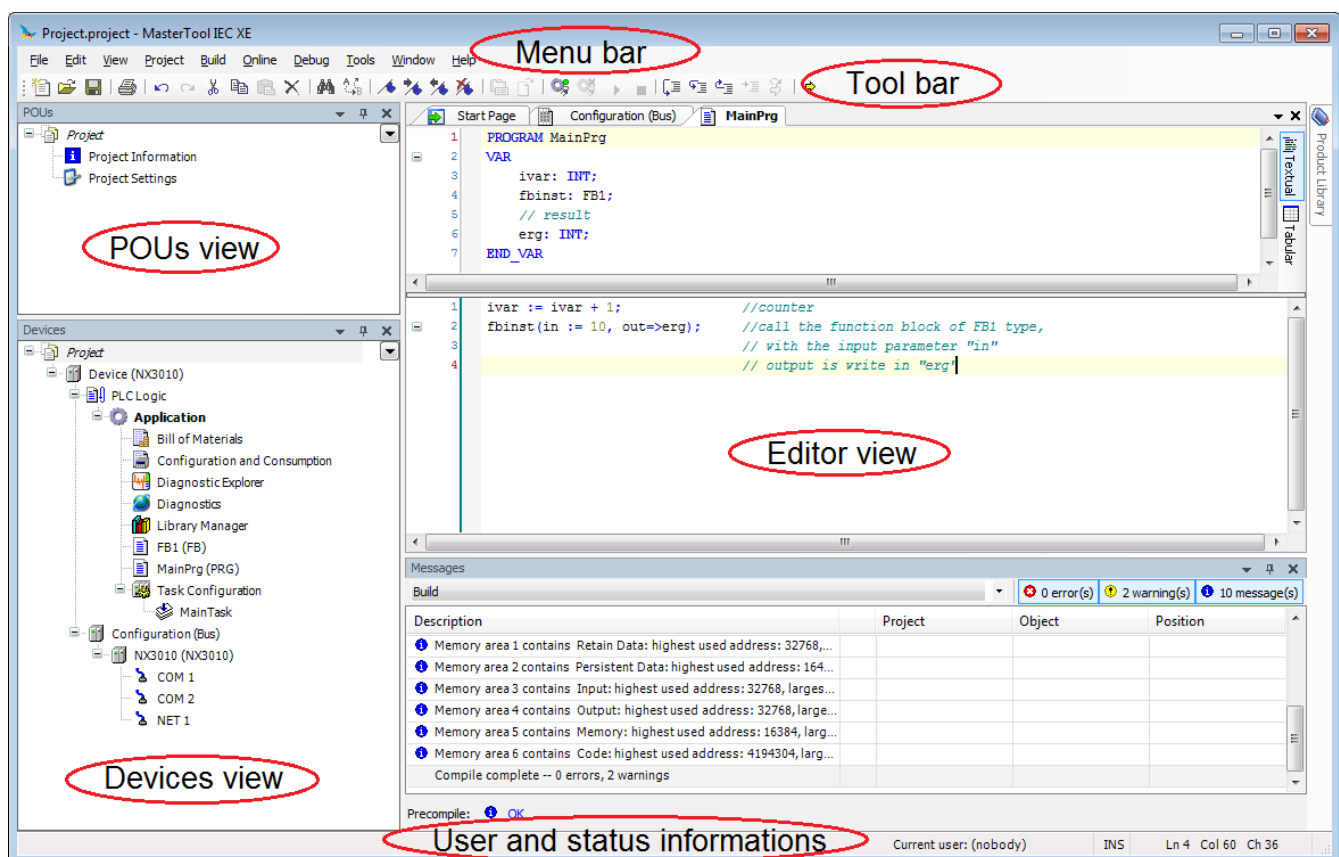


Figure 5-1. Example of the MasterTool IEC XE User Interface

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.
- *Current 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:
 -  = program running.
 -  = program stopped.
 -  = 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.

Windows, Views, Editor Windows

The windows you get displayed within or beside the user interface frame window at first sight all look the same. However, there are two types:

- Some can be docked to any margin of the frame window or alternatively can be positioned on the screen as undocked windows independently from the frame window. In addition, they can be "hidden", that is just represented by a tab in the frame border. Those windows display information, which is not depending on a single object of the project, for example Messages, Devices, POUs, Toolbox. They can be accessed via the View menu commands and also are named "views". Most views include a non-configurable toolbar with buttons for sorting, viewing, searching within the window.
- Others open when you are viewing or editing a specific project object in the respective editor. Those are displayed in a tabbed editor area or depending on the given user interface settings as

MDI windows. They cannot be "hidden" or undocked from the frame window. They can be accessed via the *Window* menu commands.
Additional types of windows or views might be added via Manufacturer specific components.

Customizing User Interface

The actual look of the user interface, which means arrangement and configuration of the particular components, depends on the following:

- Standard pre-settings for menus, keyboard functions and toolbars. Default settings are installed with MasterTool IEC XE.
- 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.

Arranging Menu Bars and Tool Bars

The menu bar is always positioned at top of the user interface, between window title bar and view windows. A toolbar can be positioned within the same area as the menu bar (fix) or as an independent window anywhere on the screen.

To re-position a bar, click with the cursor on the dotted line at the left end of the bar, keep the cursor pressed and shift the bar to the desired position.

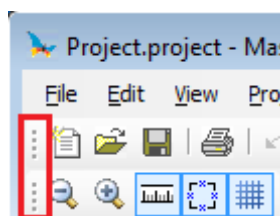


Figure 5-2. Dotted Line on Left Side of Menu and Tool Bars

NOTE: In view windows like for example POU's, Devices or Visual Element Properties a special toolbar is available, providing buttons for sorting, viewing, searching within the window. This bar cannot be configured.

Zoom

Each editor window provides a zoom function, except in the ST language editor. The zoom button (🔍) in the lower left corner of the window opens a list from where you can choose one of the zoom levels: 25, 50, 75, 100, 150, 200, and 400 percent. Notice that a printout always refers to the 100% view. The user can also define the zoom level according to his needs, he just has to enter the desired value in the corresponding field.

User Interface in Online Mode

As soon as you log in with the project, all objects which have been opened already in offline mode, automatically will be viewed in online mode.

To open an object in online mode which had not been opened already in offline mode, perform a double-click on the object entry in the POU's or Devices window or use command *Edit Object*.

If your choice is unambiguous, the object will be opened in online mode. Otherwise, for example if there are several instances of the selected object (function blocks etc.) contained in the project, a dialog named *Select Online State <object name>* will appear, where you can choose whether an instance or the base implementation of the object should be viewed and whether the object should be displayed in online or offline mode.

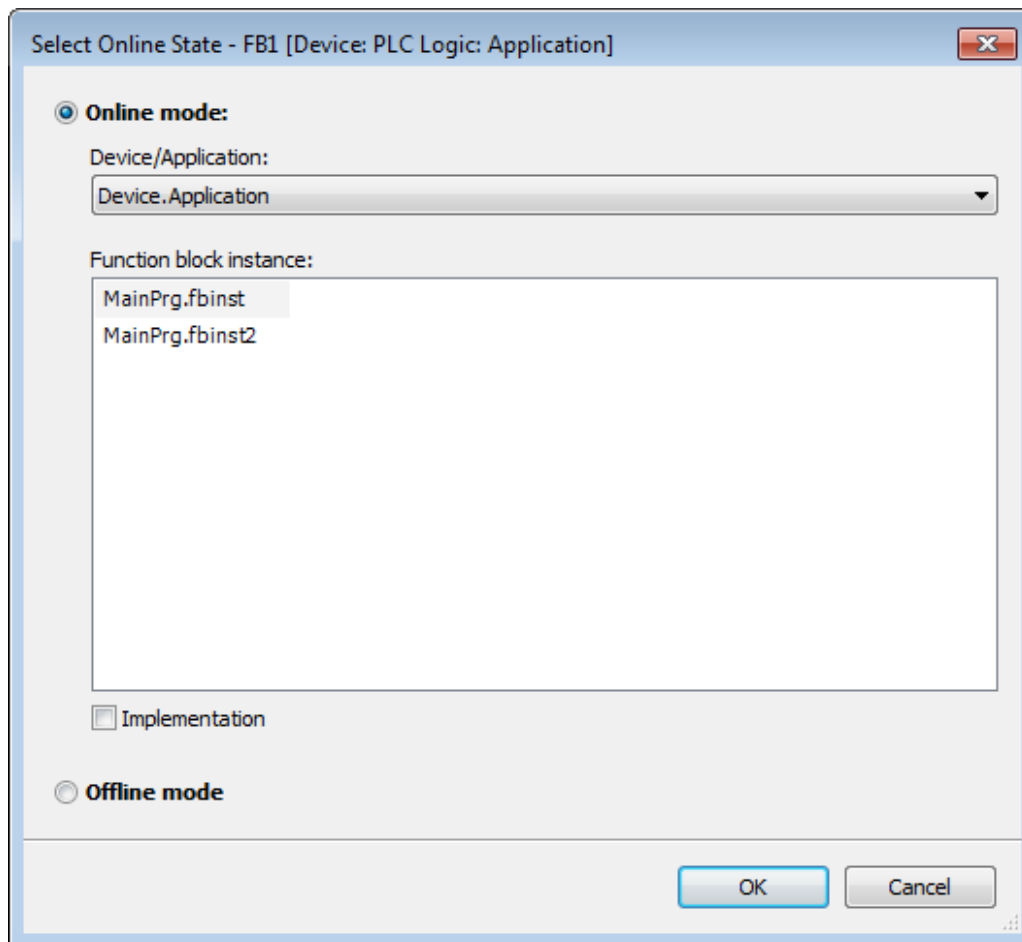


Figure 5-3. Select Online State Dialog

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, or
- 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.

Standard Menus and Commands

See in the following an overview on structure of the main menus and commands.

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

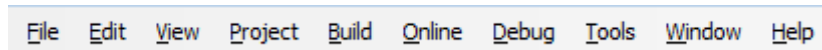


Figure 5-4. Standard Menu Bar

Standard Menus and Commands

File Menu

Commands for 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...	
	Page Setup...	
	Recent projects	
	<n><project path>	
	...	
	Exit	<Alt>+<F4>

Table 5-1. File Menu

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	<Shift>+<Ins>
	Delete	<Del>
	Select All	<Ctrl>+<A>
	Find & Replace	
	Find	<Ctrl>+<F>

	Replace	<Ctrl>+<H>
	Find Next	<F3>
	Find Next (Selected)	<Ctrl>+<F3>
	Find Previous	<Shift>+<F3>
	Find Previous (Selected)	<Ctrl>+<Shift>+<F3>
	Browse	
	Go to Definition	
	Browse for Crossed References	
	Insert File As Text...	
	Advanced	
	Overwrite Mode	<Ins>
	Go to Line	
	Make Uppercase	<Shift>+<Ctrl>+U
	Make Lowercase	<Ctrl>+<U>
	Go To Matching Bracket	
	Select To Matching Bracket	
	Bookmarks	
	Toggle Bookmark	<Ctrl>+<F12>
	Next Bookmark	<F12>
	Previous Bookmark	<Shift>+F12
	Clean Bookmarks	
	Input Assistant...	F2
	Autodeclare...	<Shift>+F2
	Next Message	F4
	Previous Message	<Shift>+F4
	Go to Source Position	

Table 5-2. Edit Menu

View Menu

Commands for activating the particular standard views, that is getting them displayed in a window in the user interface.

Symbol	Command	Shortcut
	POUs	<Alt> + 0
	Devices	<Alt> + 1
	Messages	<Alt> + 2
	Element Properties - for SFC element - for Visualization element	
	Product Library	
	Toolbox	
	Watch	
	Watch <n>* * n= 1,2,3,4	
	Watch All Forces	
	Breakpoints	
	Call Stack	

	Cross Reference List	
	Start Page	
	Full Screen	<Ctrl>+<Shift>+<F12>
	Properties...	

Table 5-3. View Menu

Project Menu

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

Symbol	Command	Shortcut
	Add Object...	
	External File...	
	DUT...	
	Global Variable List...	
	PID Control...	
	POU...	
	Action...	
	Method...	
	Property...	
	Transition...	
	POU for Implicit Checks...	
	Trace...	
	Persistent Variables...	
	Library Manager...	
	Task Configuration...	
	Task...	
	Add Device...	
	Add Folder...	
	Edit Object	
	Edit Object with...	
	Set Active Application	
	Project Information...	
	Project Settings...	
	Project Update...	
	Document...	
	Compare...	
	User Management	
	Logon...	
	Logoff	
	Permissions...	

Table 5-4. Project Menu

Build Menu

Commands for building the project that is for doing a pre-compilation run including a syntactical check. In addition, commands for deleting the last compile information (*Clean* command) which is of meaning for online change and for offline code generation.

Symbol	Command	Shortcut
	Generate code	
	Clean	
	Clean all	

Table 5-5. Build Menu

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	
	Logoff current online user	
	Download	
	Online Change	
	Source download to connected device	
	Cluster Basic Configuration	
	Reset warm	
	Reset cold	
	Reset origin	
	Simulation	

Table 5-6. Online Menu

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...	
	Toggle Breakpoint	<F9>
	Step Over	<F10>
	Step Into	<F8>
	Step Out	<Shift>+<F10>
	Run to Cursor	
	Set next statement	
	Show next statement	
	Write values	<Ctrl>+<F7>
	Force values	<F7>

	Unforce values	<Alt>+<F7>
	Add All Forces to the Watchlist	
	Display Mode	
	Binary	
	Decimal	
	Hexadecimal	

Table 5-7. Debug Menu

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...	
	Options...	

Table 5-8. Tools Menu

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 Editor	<Ctrl>+<F4>
	Close All Editors	
	Window Layout Reset	
	New Horizontal Tab Group	
	New Vertical Tab Group	
	Float	
	Dock	
	Auto Hide	
	Next Pane	<F6>
	Previous Pane	<Shift>+<F6>
Editor Icon	<n> <window title> (application path)]...	
	Windows...	

Table 5-9. Window Menu

Help Menu

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

Symbol	Command	Shortcut
	Content	<Ctrl>+<<Shift>+<F1>
	Index	<Ctrl>+<<Shift>+<F2>

	Search	
	Contact Support	
	Update Software License...	
	Altus Home Page...	
	About...	

Table 5-10. Help Menu

Bus Editor Menu

Commands related to customization and best bus editor preview (Configuration (Bus)).

Symbol	Command	Shortcut
	Zoom out	
	Zoom in	
	Rulers	
	Ports	
	Grid	

Table 5-11. Graphic Editor Menu

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.

In order to that be possible, the user must access the project under development in the MasterTool IEC XE software and click on the Devices Tree, placed at the program left. Two clicks must be executed on the *Device* item and the *Files* tab selected, as showed on Figure 5-5.

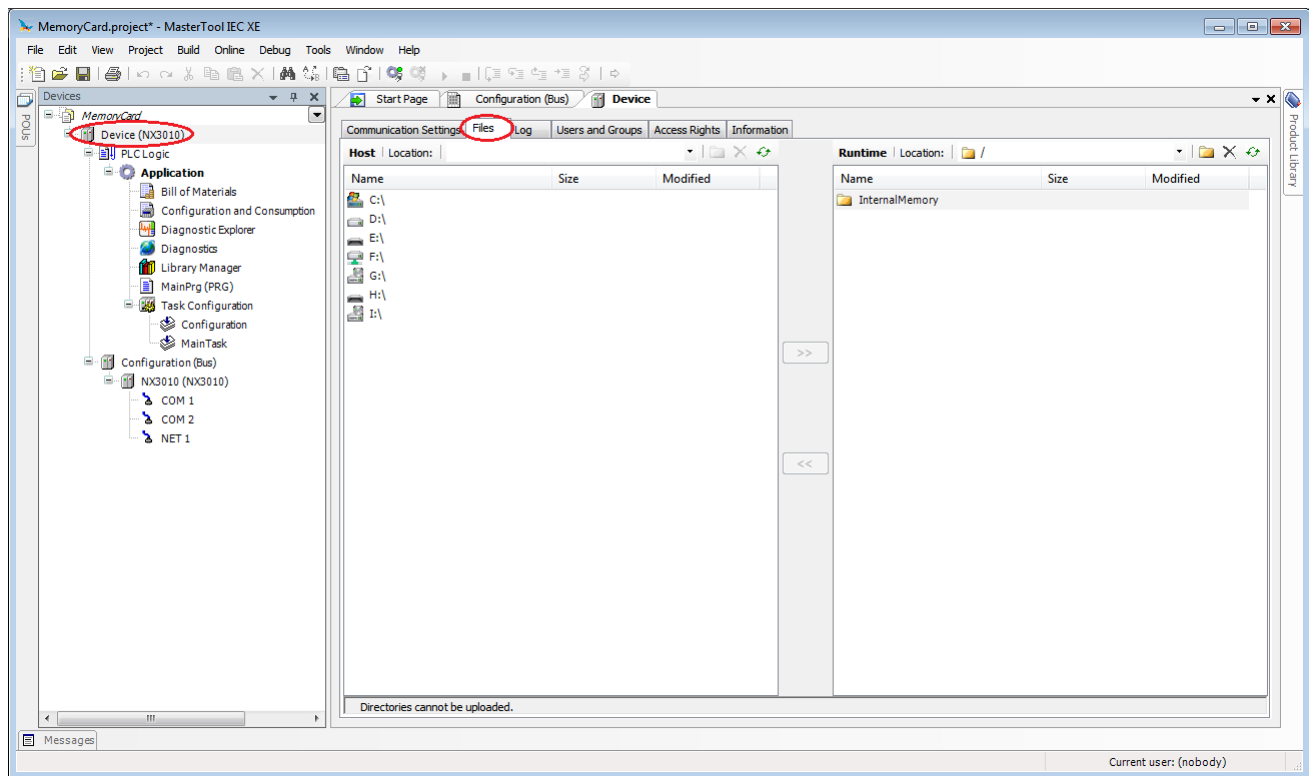


Figure 5-5. User Files Access

Then it is just to select the desired file in the left column and press “>>” button, placed at the screen center. The transfer time varies according to the file size, it may take a few seconds.

The user does not need to be in Run Mode or connected to the CPU for transferring, as it has the capacity to connect itself automatically when the user starts the transference.

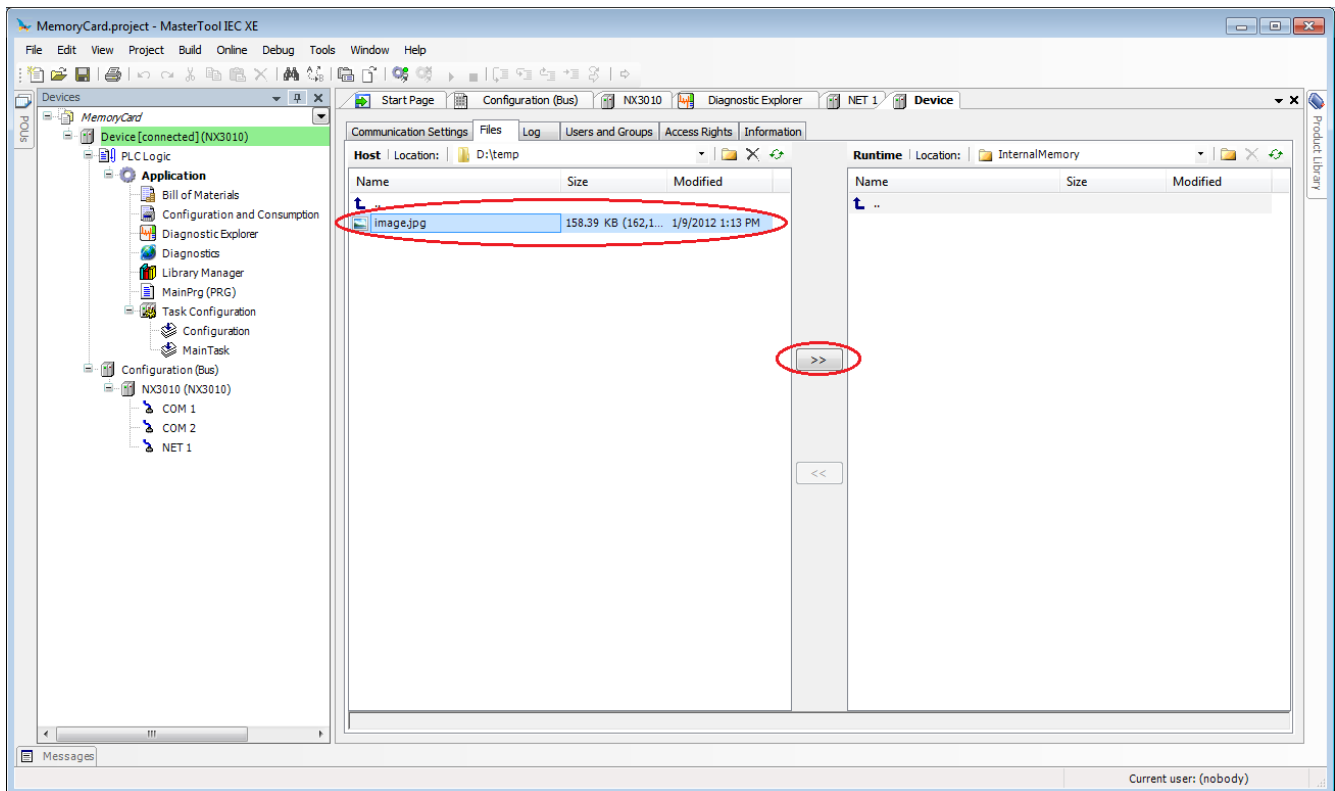


Figure 5-6. Transferring Files

In case it is necessary to transfer documents from CPU to the PC, where the MasterTool IEC XE software is installed, the user must follow a procedure very similar to the previously described, as the file must be selected from the right column and the button “<<” pressed, placed on the center of the screen.

Furthermore, the user has some operation options in the storing files area, which are:

- New folder  : allows the creation of a new folder in the user memory area.
- Exclude item  : allows the files excluding in the folders in the user memory area.
- Update  : allows the file updating, on the MasterTool IEC XE screen, of the files in the user memory area.

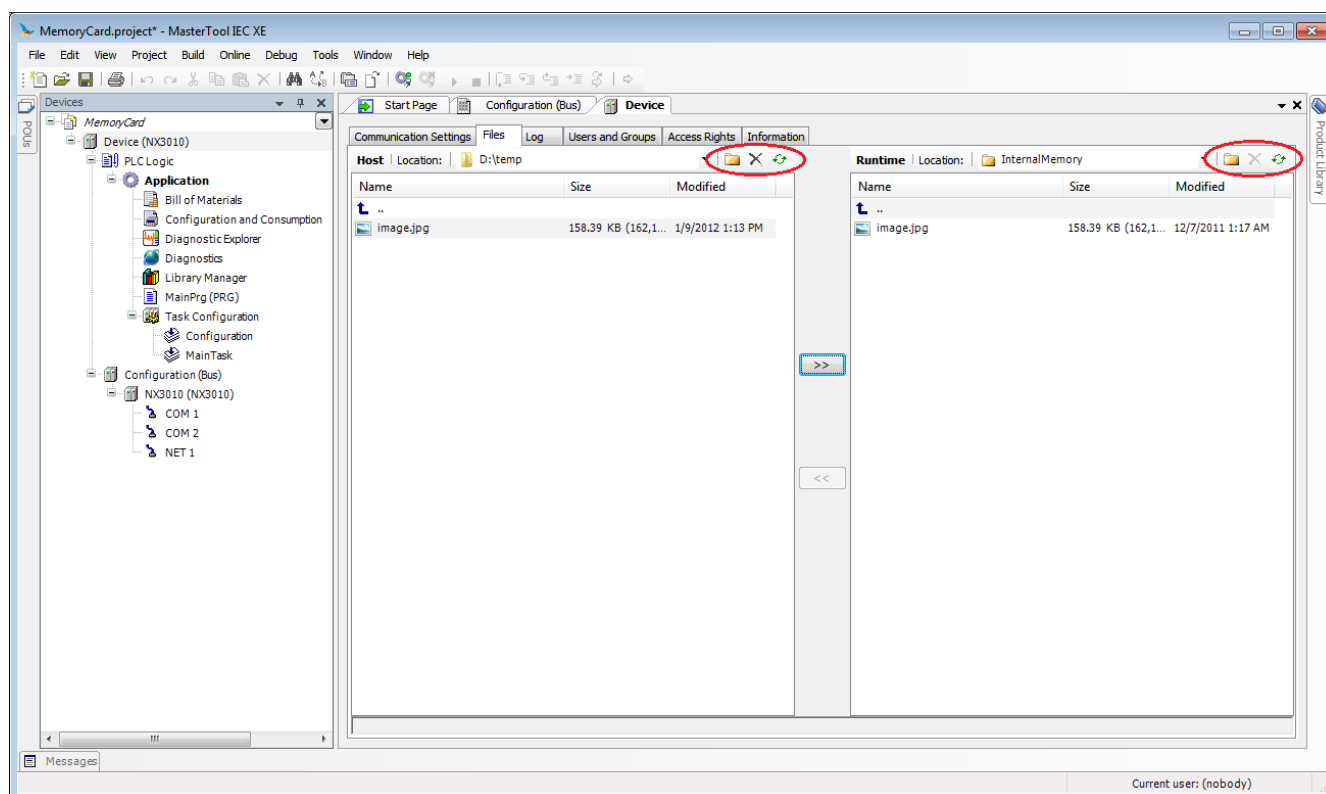


Figure 5-7. Options for Operation of the File Storage Area

For further information regarding this product feature, see the Manual of the used CPU.

6. User and Access Right Management

Provide functions for defining user accounts and configure the access rights within a project. Notice that also device specific user management might be supported for controlling the user's access rights on the PLC file system and objects during runtime.

The rights to access project objects via specified actions are assigned only to user groups, not to a single user account. Therefore, each user must be member of a group.

User Management

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

Automatically there is always a group *Everyone* and by default primarily each defined user or other groups are members of this group. Thus, each user account at least automatically is provided with defined default settings. Group *Everyone* cannot be deleted, just renamed, and no members can be removed from this group.

Also automatically there is always a group *Owner* containing one 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.

Access Right Management

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

In a new project basically all rights are not yet defined explicitly but set to a default value. This default value usually is: "granted".

In the further run of working on the project each right can be explicitly granted or denied and set back to default. The access right management of a project is done in the *Permissions* dialog or - for object access rights - in the *Access control* dialog, which is part of the object *Properties* dialog.

Access rights on objects get "inherited". If an object has a "father" object (example: if an action is assigned to a program object, that is inserted in the structure tree below the program, then the program is the "father" of the action object), the current rights of the father automatically will become the default settings of the child. Father-child relations of objects concerning the access rights usually correspond with the relations shown in the POU's or Devices tree and are indicated in the *Permissions* dialog by the syntax "<father object>.<child object>". Example: Action ACT is assigned to the object MainPrg (POU). So in the POU's window ACT is shown in the objects tree indented below MainPrg. In the *Permissions* dialog ACT is represented by "MainPrg.ACT" indicating that MainPrg is the "father" of ACT. If the "modify" right would be denied explicitly for

MainPrg and a certain user group, the default value of the "modify" right for ACT automatically also would be "denied".

Users and Groups

For devices supporting an "online user management", this dialog is provided on a tab of the *Device* dialog (Device Editor). It allows for setting up user accounts and user groups, which in combination with the Access Rights management serves to control the access on objects on the PLC in online mode.

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.

Using the Configuration Dialog

Basically, the handling of the 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.

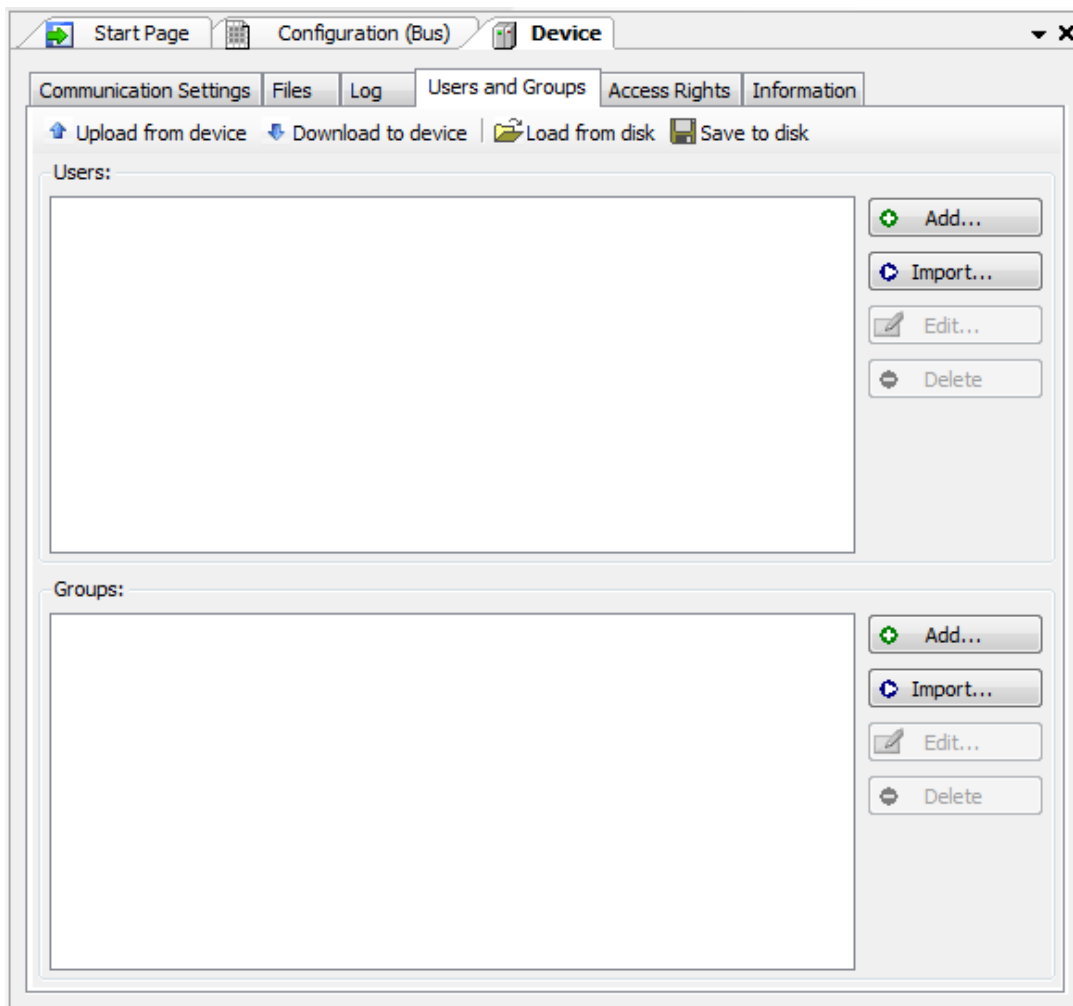
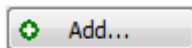


Figure 6-1. Device Dialog, Users and Groups

Users

The following buttons are available for setting up user accounts:



: 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.

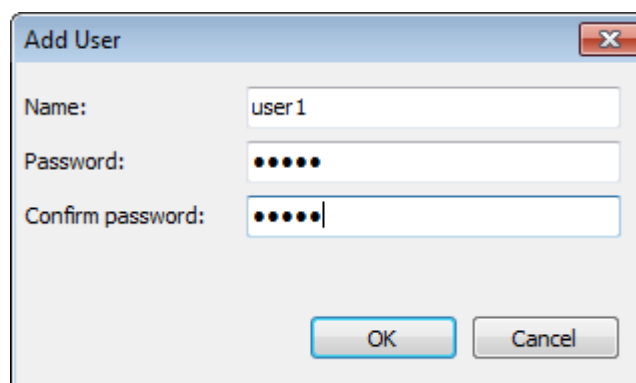
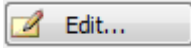


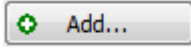
Figure 6-2. Add User Dialog

: 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.

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

: The currently selected user account will be deleted.

Groups

: 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.

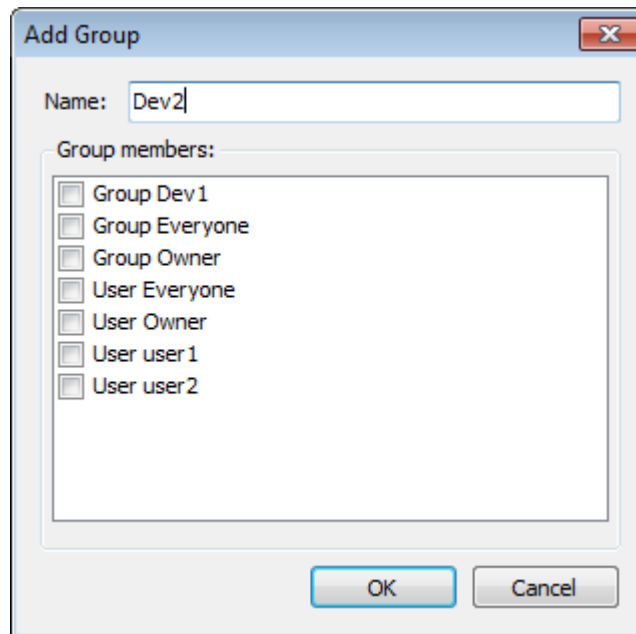
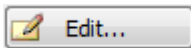


Figure 6-3. Add Group Dialog

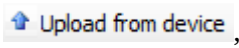
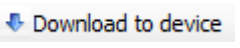
: The dialog *Import Groups* opens where the groups currently defined in the project user management are listed. You can select one or several entries and confirm with OK to get them integrated in the groups list of the device-specific user management.

: 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.

: The currently selected group can be deleted.

Applying and Storing the Current Configuration

See the respective buttons in the top bar of the dialog.

 ,  : The current user management configuration must be downloaded to the device to get it effective. The configuration currently applied on the device can be uploaded into the configuration dialog.

 **Save to disk**,  **Load from disk**: The current configuration can be stored in a xml-file (*.dum) 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 "*.dum", which means "device user management" files).

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

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 serves to grant and deny the currently defined user groups certain permissions, thus defining the users' access rights on files or objects (like for example an application) on the PLC during runtime.

Notice that these permissions can only be assigned to groups, not to particular users. For this reason, a user must be defined as member of a group. The configuration of users and groups is done in the *Users and Groups* tab of the device editor.

Figure 6-4 shows the permission to add and remove children to/from the PLC Logic object granted for user group "main_control".

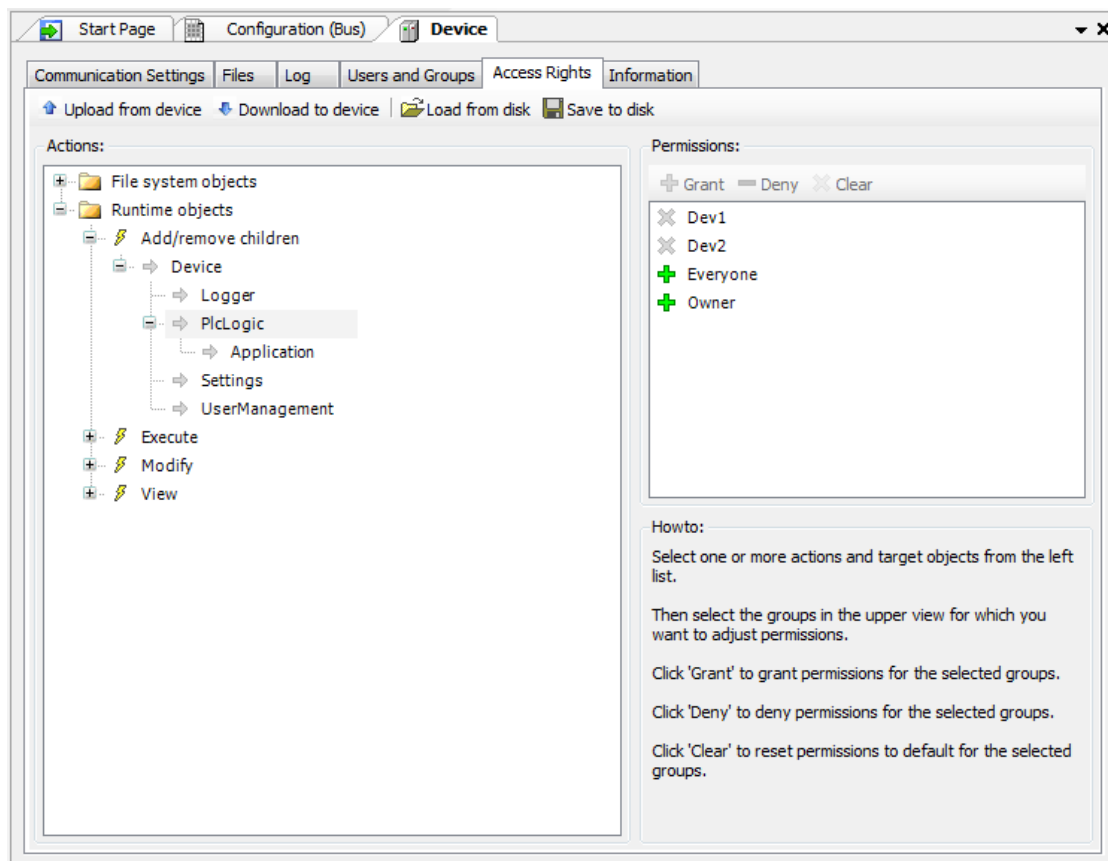


Figure 6-4. Device, Access Rights

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.

Defining the Access Permissions

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 *Actions* window, then select the desired group in the *Permissions* window, and then click on the *Grant* or *Deny* button (also in the *Permissions* window).

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

Actions

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:

Object categories

Top-level, for structuring purposes, there are two "folders" for File system objects and Runtime objects.

Action types

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")

Below each action type node finally find the "targets" (objects) of the action (for example ⇒ Device).

These object entries are displayed in a tree structure mapping the device tree or the file system structure.

NOTE: 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.

Permissions

This field shows the currently defined user groups. Before each group one of the following icons indicates the currently assigned permission concerning the target, which is currently selected in the *Actions* window:

: The action(s) currently selected in the *Actions* window are granted for the group.

: The action(s) currently selected in the *Actions* window are denied for the group.

: Currently there is no explicit access right definition for the action(s) currently selected in the *Actions* window.

If currently multiple actions are selected in the *Actions* window, which do not have unique settings referring to the currently selected group, no icon will be displayed.

Button bar: After having selected the desired object(s) below the desired action in the *Actions* window and having selected the desired group in the *Permissions* window, one of the following buttons can be used:

 **Grant** : Explicit granting access permission.

 **Deny** : Explicit denying access permission.

 **Clear** : The currently granted access right for the action(s) currently selected in the *Actions* window will be deleted, that is set back to the default.

Applying and Storing the Current Configuration

See the respective buttons in the top bar of the dialog.

 **Upload from device** ,  **Download to device** : The currently configured Access Rights definitions must be downloaded to the device to get it effective. The configuration currently applied on the device can be uploaded into the configuration dialog.

 **Save to disk** ,  **Load from disk** : The current configuration can be stored in a xml-file (*.dar) 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 “*.dar”, which means “device access rights” files).

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

Status

This dialog is provided on a tab of the *Device* dialog (Device Editor). It shows status information (for example "RUN", "STOP") and specific diagnosis messages from the respective device; also on the used card and the internal bus system.

Information

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.

If available for the current device, the following general information will be displayed: Name, Vendor, Type, Version number, Order Number, Description, and Image.

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.

File Menu

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

Available commands:

- New Project...
- Open Project...
- Close Project
- Save Project
- Save Project As...
- Project Archive
- Source upload...
- Source download...
- Print...
- Page Setup...
- Export to CSV
- Import to CSV
- Document
- Recent Projects
- Exit

New Project

Symbol: 

Default Shortcut: <CTRL>+<N>

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

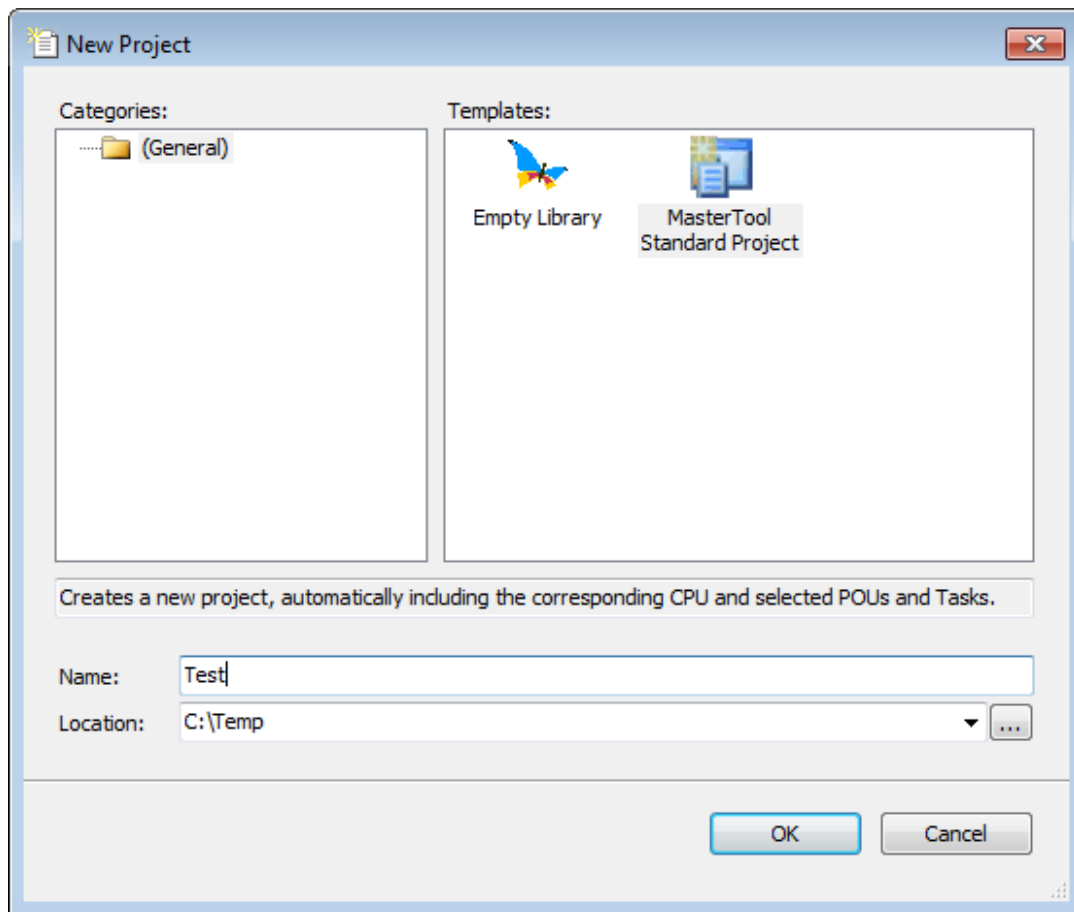


Figure 7-1. New Project

- **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.
- **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 *MasterTool Standard Project* (.project) that automatically provides the CPU and inserted devices, POU's and tasks according to user choices. There is also a model for Empty Library (.library).
- **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 "Unbekannt1"). 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.
- **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.
- **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.

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.

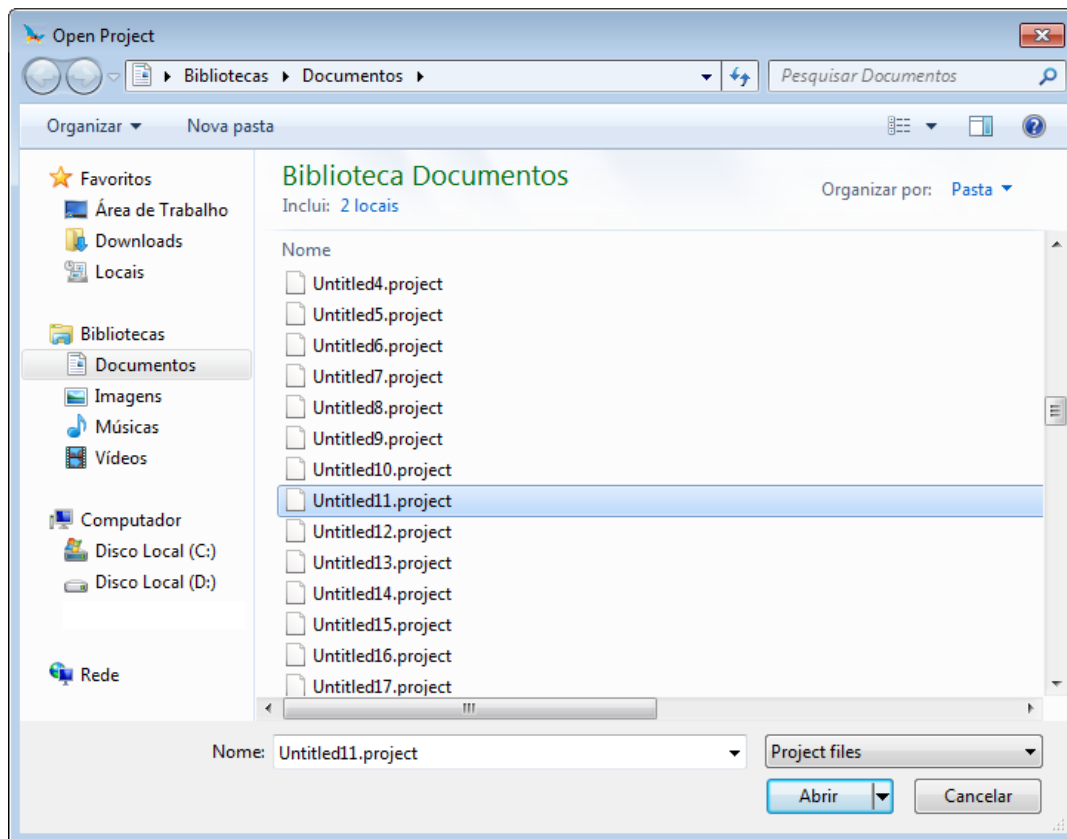


Figure 7-2. Open Project

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

Another Project is Open

You will be asked whether it should be saved and closed.

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**.

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.

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.

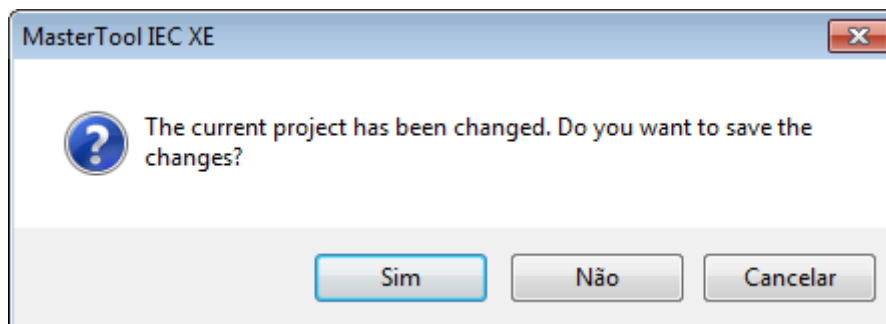


Figure 7-3. Project Alterations

For terminating MasterTool IEC XE use command Exit.

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.

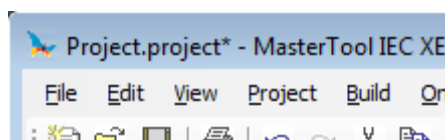


Figure 7-4. Title bar, Modified Project

In case the project is write-protected, the following message dialog (Figure 7-5) will be opened:

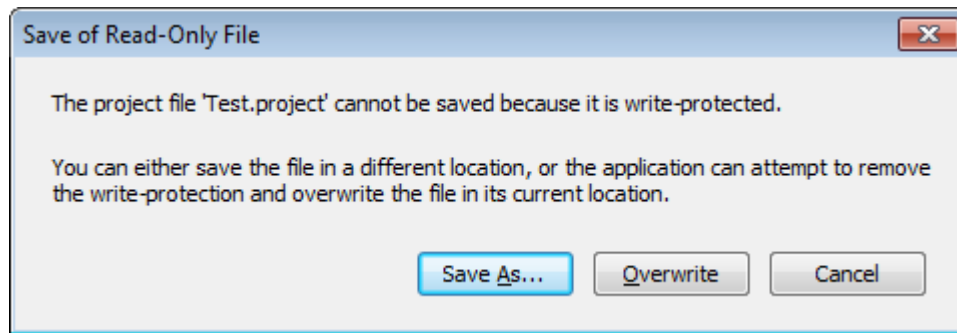


Figure 7-5. Message Dialog, Project Write-Protected

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

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.

Project Archive: 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.

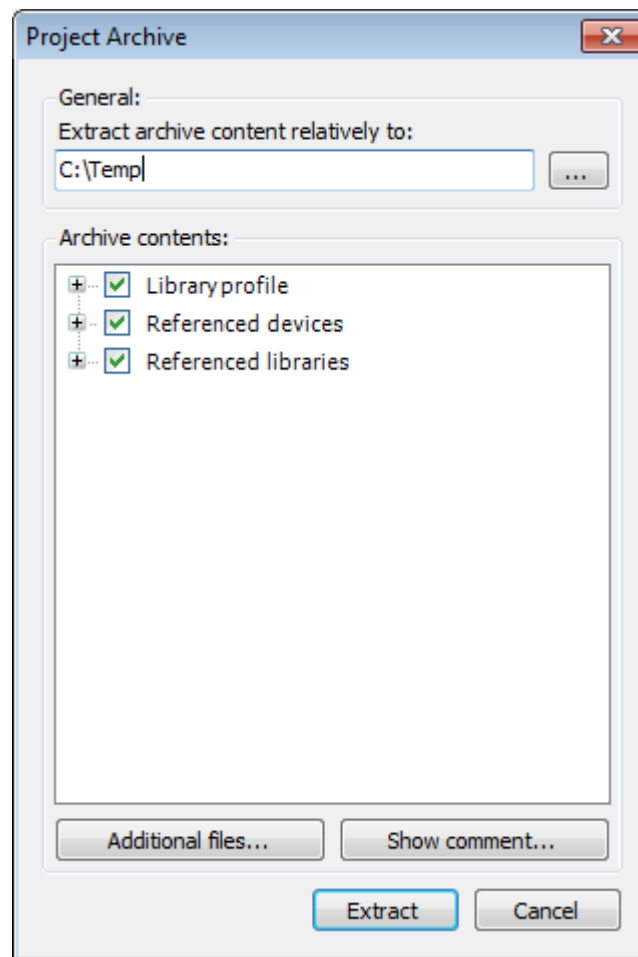


Figure 7-6. Project Archive

Under *Extract archive content relatively to:* 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 *Additional files...* there will pop up the following dialog box:

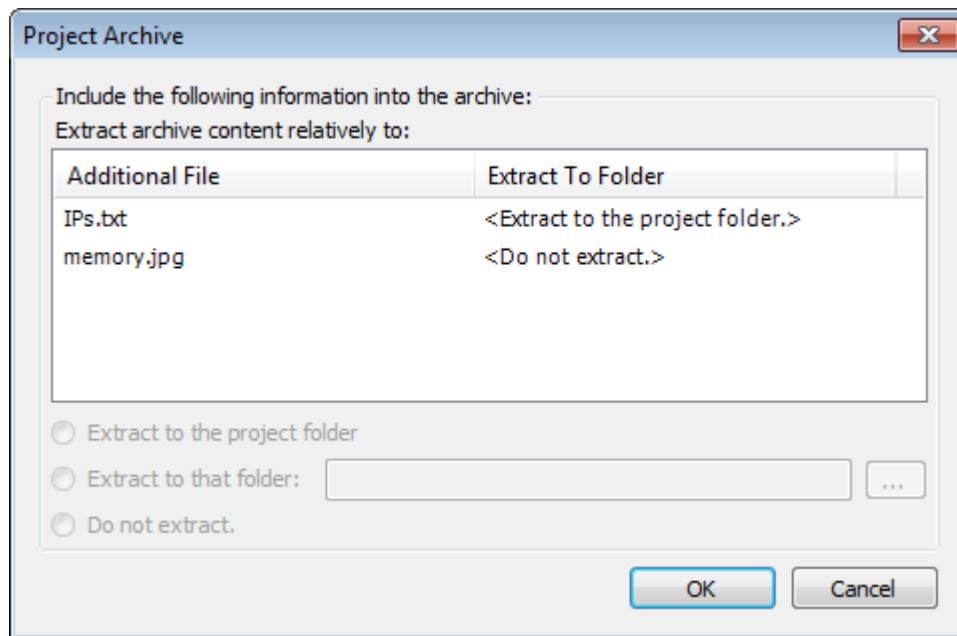


Figure 7-7. Additional Files Dialog

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 it by a click on its name. Afterwards you have to choose one of the three options provided below:

- *Extract to 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 to that 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.

After adjusting the extract properties of the additional files you may exit the Additional file dialog pressing OK.

A click on *Show comment...* will display the comment eventually given on the archived project by its author. If no comment is contained in the archive, a corresponding error message will be displayed instead.

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 that ought's to be extracted is named in the same way as a file already existing in the target directory, a warning message will be displayed and you will be asked whether you want to replace the local file or not. You can make your decision become valid for every further name mismatch by activating the option  Apply to all items and files.
- *Cancel:* to abort the operation; the archive will not get extracted.

Project Archive: 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 give forward the set of all project relevant files as the archive can be unpacked easily by use of the command *Extract Archive*.

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

After executing the command, the dialog box *Project Archive* opens.

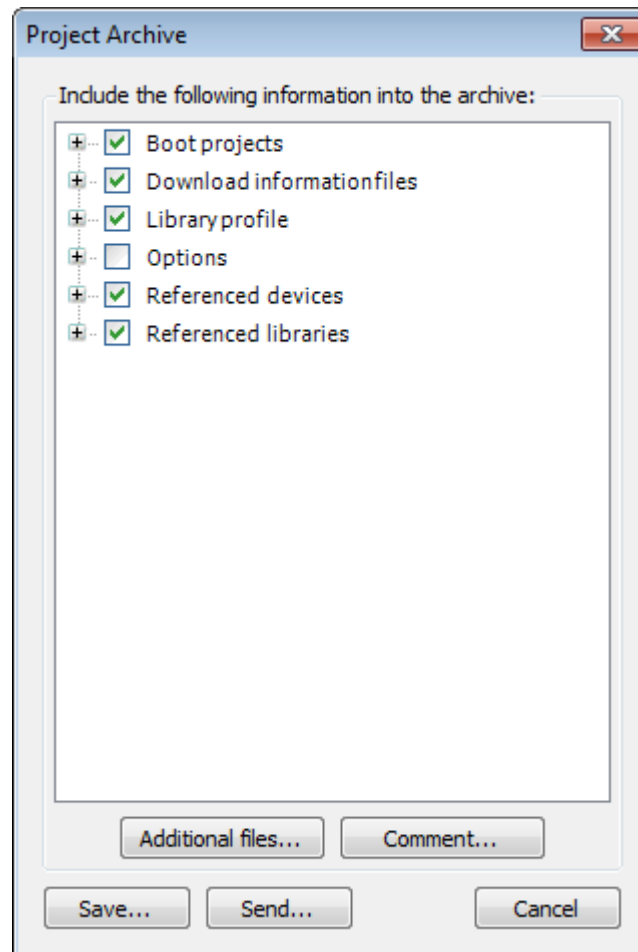


Figure 7-8. Project Archive Dialog

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 an category are activated, its mark will appear as .

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

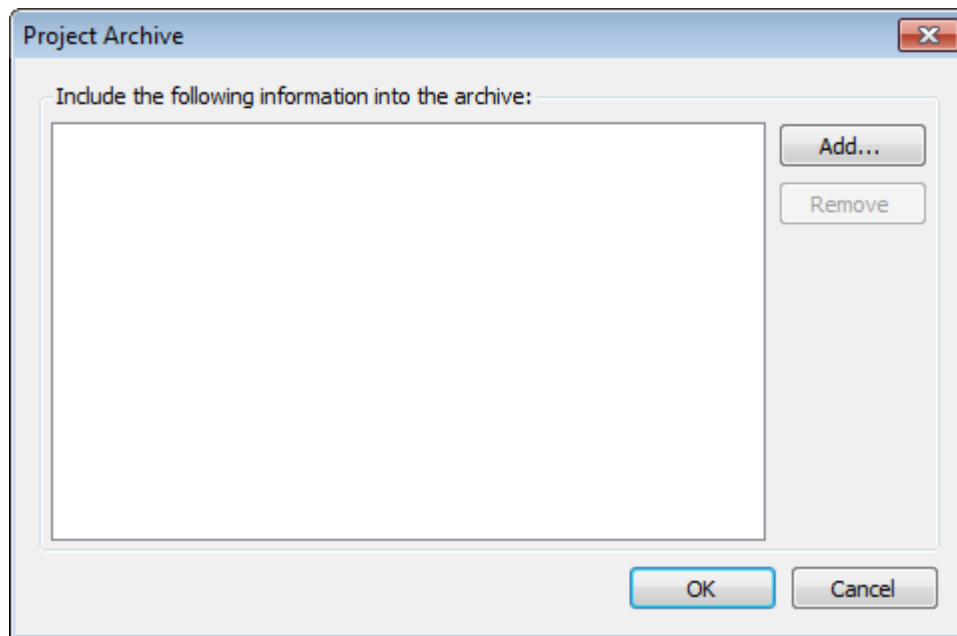


Figure 7-9. Additional Files

Press the button *Add* 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 *Additional files...* dialog. To remove a file from this list, select it by a click and press the button *Remove*. After the list of selected files fits to 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 will be imported, your comment can be accessed in the corresponding dialog via *Show comment....*

After carrying out the desired enlistment choose one of the following items of the main dialog:

- **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.

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.

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*.

Print

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. For further information, see **Export to CSV**.

Page Setup

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

Recent Projects

Use this command to select from a list of the most recently opened projects that one you want to reopen. The number of projects, which are kept in this list, is defined in the *Load and Save* options (*Tools* menu).

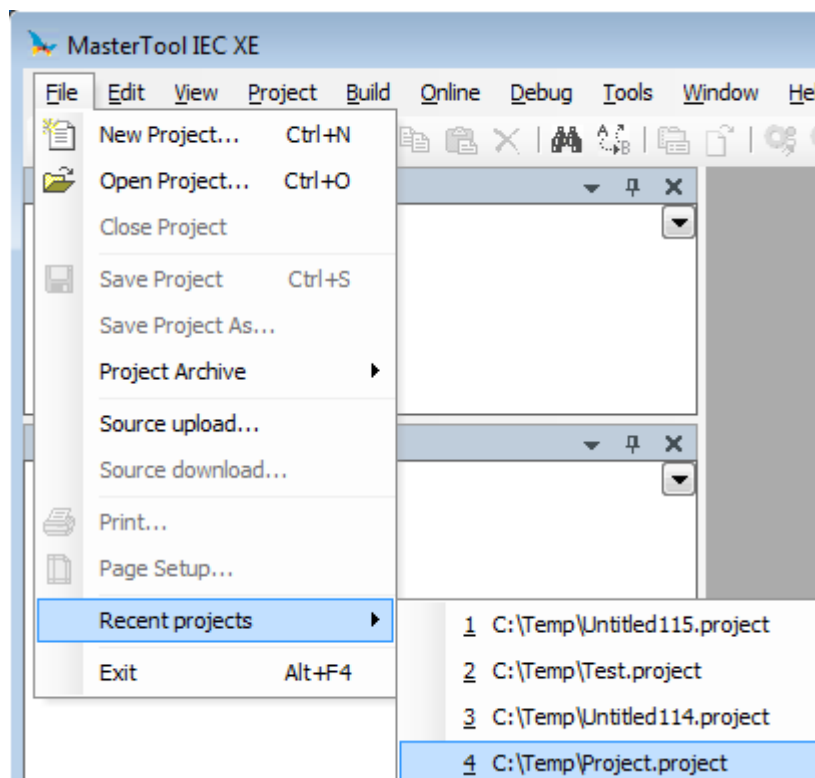


Figure 7-10. Recent Projects List

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.

Edit Menu

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

Available commands:

- Undo
- Redo
- Cut
- Copy
- Paste
- Delete
- Select All
- Find & Replace
- Browse
- Insert File As Text
- Advanced
- Bookmarks
- Input Assistant...
- Auto Declare...
- Next Message
- Previous Message
- Go to Source Position

Undo/Redo

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

Available commands:

- Undo
- Redo

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 POU's, data types, visualizations and global variables.

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

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.

Clipboard

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

Available commands:

- Cut
- Copy
- Paste
- Delete

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 POU's 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.

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 POU's 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.

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.

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.

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.

Find/Replace

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

Available commands:

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

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.

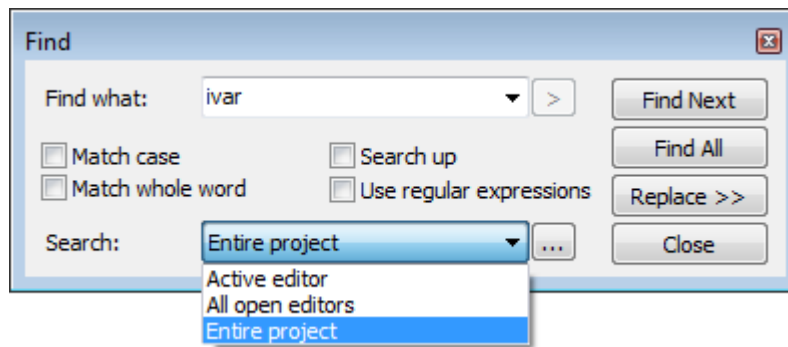


Figure 7-11. Find Dialog

Find what: Insert here the string to be searched. The selection list available via button  will be filled with strings, which have been already searched since the last start of MasterTool IEC XE.

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 .

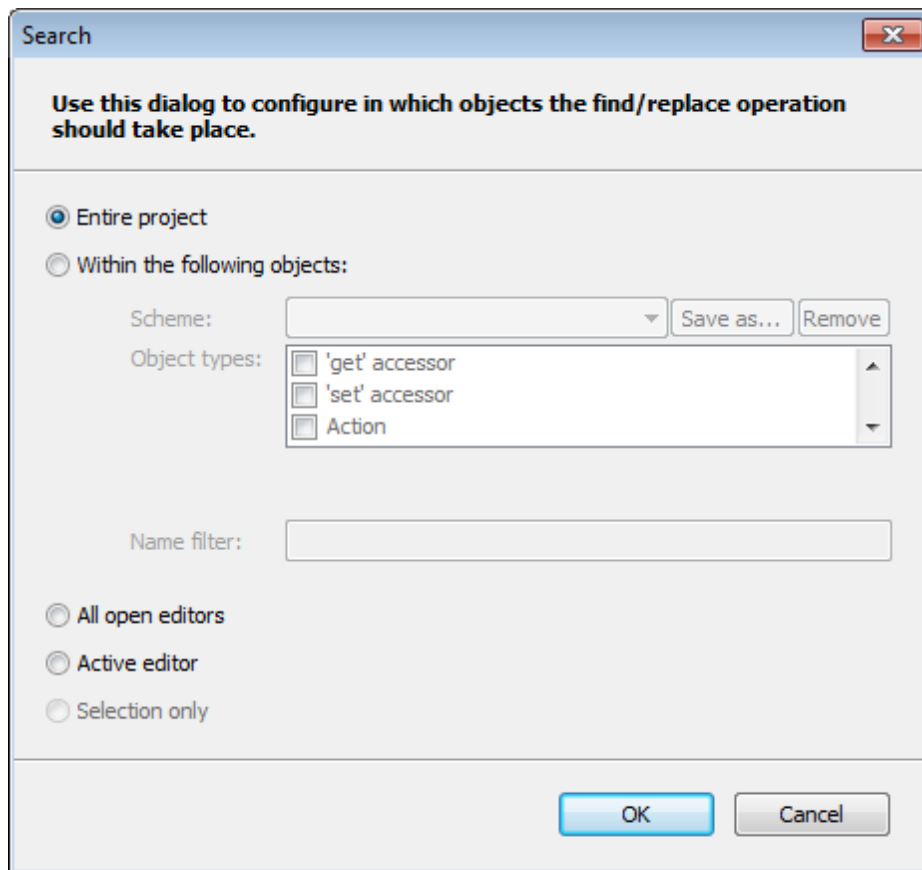


Figure 7-12. Search Dialog

- *Entire project*: All editable places within all project objects 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.

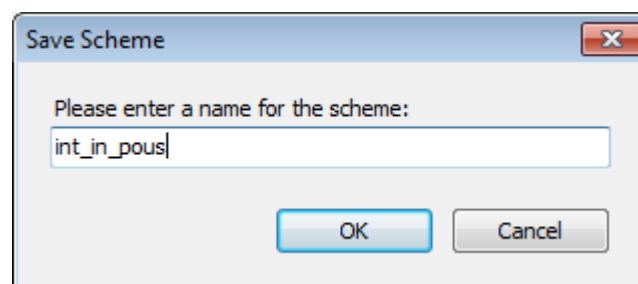


Figure 7-13. Save Scheme Dialog

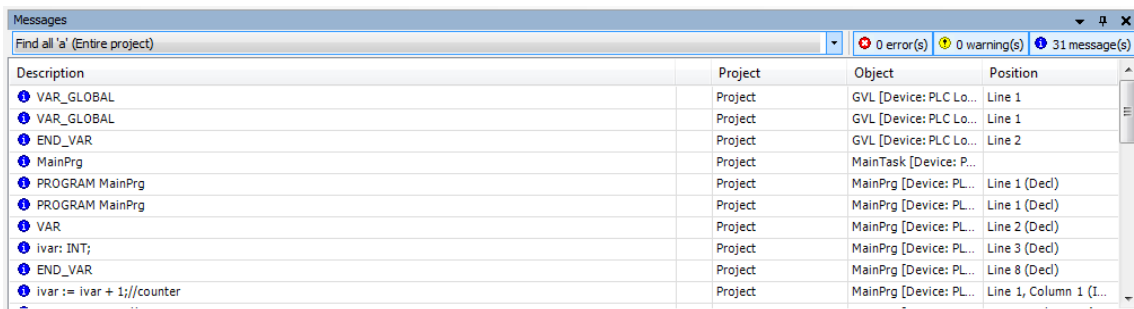
- *All open editors*: All editors currently opened in a window will be searched.

- *Active editor*: Just the editor where currently the cursor is placed, will be searched.
- *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.



The screenshot shows a 'Messages' window with a search bar containing 'Find all 'a' (Entire project)'. The status bar at the top right indicates '0 error(s)', '0 warning(s)', and '31 message(s)'. The main table lists search results with columns: Description, Project, Object, and Position.

Description	Project	Object	Position
VAR_GLOBAL	Project	GVL [Device: PLC Lo...	Line 1
VAR_GLOBAL	Project	GVL [Device: PLC Lo...	Line 1
END_VAR	Project	GVL [Device: PLC Lo...	Line 2
MainPrg	Project	MainTask [Device: P...	
PROGRAM MainPrg	Project	MainPrg [Device: PL...	Line 1 (Ded)
PROGRAM MainPrg	Project	MainPrg [Device: PL...	Line 1 (Ded)
VAR	Project	MainPrg [Device: PL...	Line 2 (Ded)
ivar: INT;	Project	MainPrg [Device: PL...	Line 3 (Ded)
END_VAR	Project	MainPrg [Device: PL...	Line 8 (Ded)
ivar := ivar + 1; //counter	Project	MainPrg [Device: PL...	Line 1, Column 1 (I...

Figure 7-14. 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.

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.

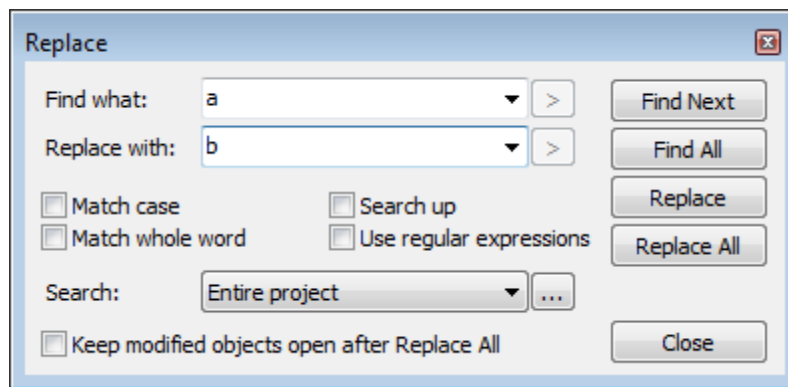


Figure 7-15. Replace

*Find Next***Symbol:**

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.

*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.

*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.

*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

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
- Browse Cross References

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.

Browse 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.

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.

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

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.

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.

Make Lowercase

Default Shortcut: <CTRL> + <U>

This command will set the currently marked text to lowercase.

Make Uppercase

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

This command will set the currently marked text to uppercase.

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.

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.

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
- Bookmark Next
- Previous Bookmark
- Clear Bookmarks

Toggle Bookmark

Symbol: 

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

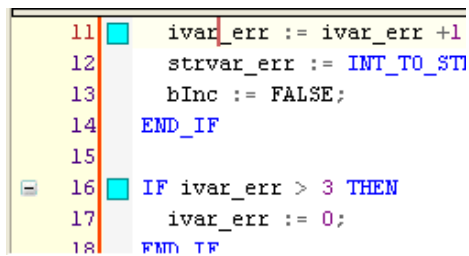


Figure 7-16. Bookmarks in ST Editor

Next Bookmark

Symbol:

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

Previous Bookmark

Symbol:

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

Clear Bookmarks

Symbol:

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

Input Assistant

Symbol:

Default Shortcut: <F2>

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

The items are sorted by Categories. For the currently selected category, the available items and their respective data type are displayed in the Items window. 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").

NOTE: 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).

If option *Show Documentation* is activated, the dialog will be extended by the *Documentation* field. If the selected element is a variable and an address is assigned to this variable and a comment has been added at its declaration, those will be displayed here.

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.

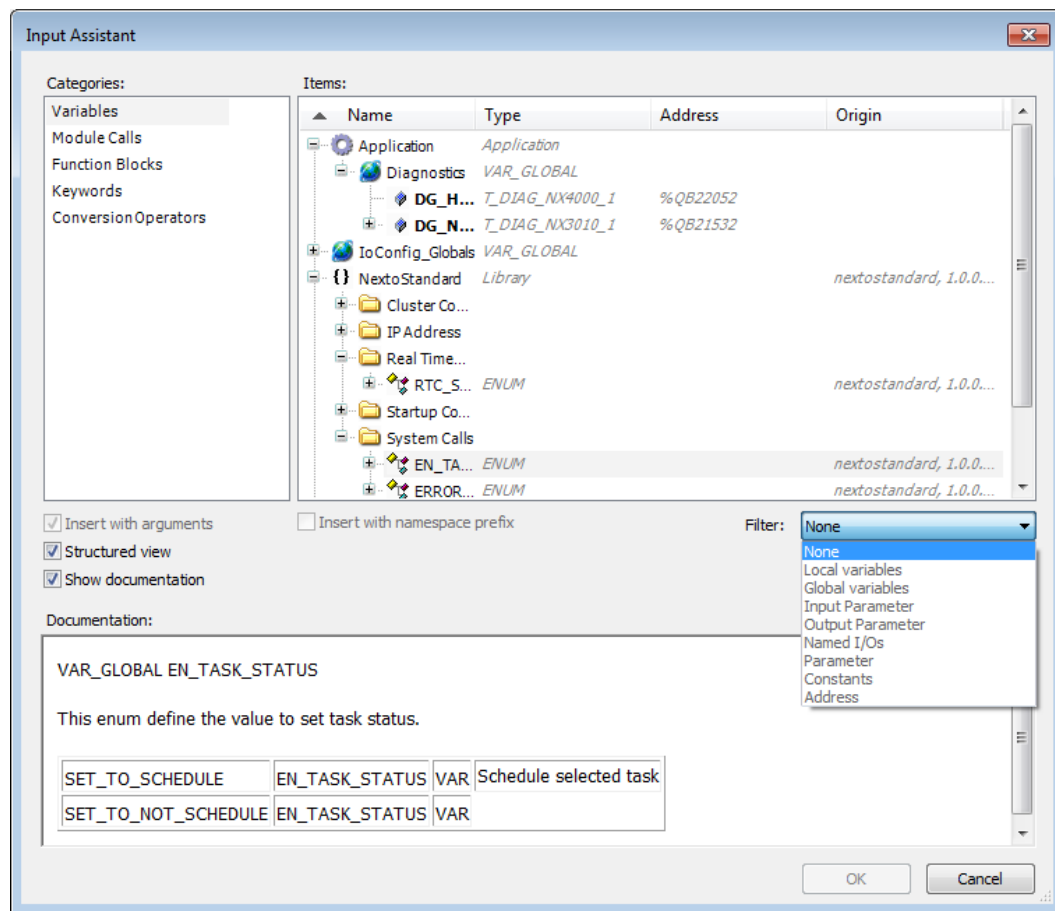


Figure 7-17. Input Assistant Dialog

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.

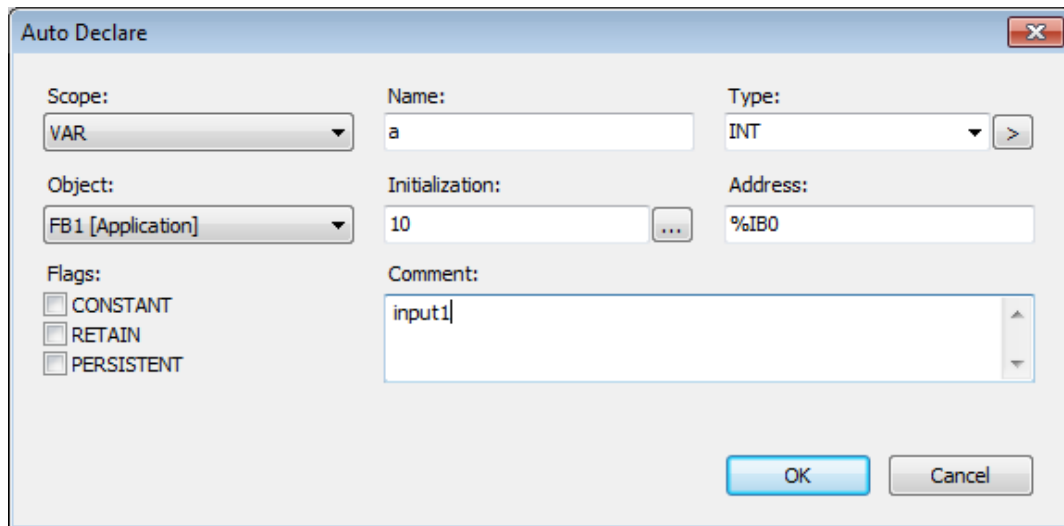


Figure 7-18. Dialog Box for Declaration of Variables

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 "b" of type "INT" and address "%IB0" -> declaration: "b 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 you are dealing with a constant or a remanent variable. The respective attribute will be added to the keyword VAR, for example "VAR CONSTANT" initiating the declaration part for the variable. The PERSISTENT option only will be available if a Persistent Variables list exists.

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:

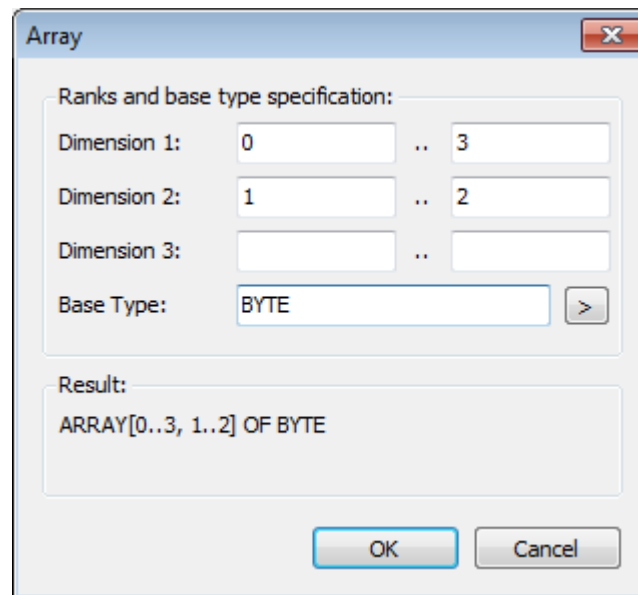


Figure 7-19. ARRAY

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.

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.

View Menu

The "view" window type can be managed by commands from this menu. These commands open the corresponding windows as the user's choice.

Available commands:

- POU's
- Devices
- Messages
- Element Properties
- Product Library

- Toolbox
- Watch
- Breakpoints
- Call Stack
- Cross Reference List
- Start Page
- Full Screen
- Properties...

Default Navigators

Provides the following commands of View menu, which will be visible as windows in the user interface:

- POU's
- Devices

By default each view in the upper right corner has a button  for opening a menu with the following commands and options:

- *Open in editor*: Opens the object in the appropriate editor.
- *Find object*: Opens where you can search the objects tree for a POU name.
 - *Find what*: enter the search string (may be a single character) and activate the desired options.
 - *Match whole name*: The search will provide all object names, which exactly match the sequence of characters of the search string.
 - *Match case*: The search will provide all object names, which match the search string exactly concerning the use of upper and lower letters.
 - *Match prefix*: The search will provide all object names starting with the search string.
 - *Match Substring*: The search will provide all object names containing the search string.

The found objects will be displayed with Name and Path in the Result window.

- *Sort by type, Sort by name*: The objects get sorted by type or name, and alphabetically.
- *Sort ascending, Sort descending*: The above defined sorting is arranged in ascending or descending order.

POUS

Symbol: 

Default Shortcut: <ALT> + <0>

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

Devices

Symbol: 

Default Shortcut: <ALT> + <1>

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

Messages

Symbol: 

Default Shortcut: <ALT> + <2>

This command opens the *Messages* window.

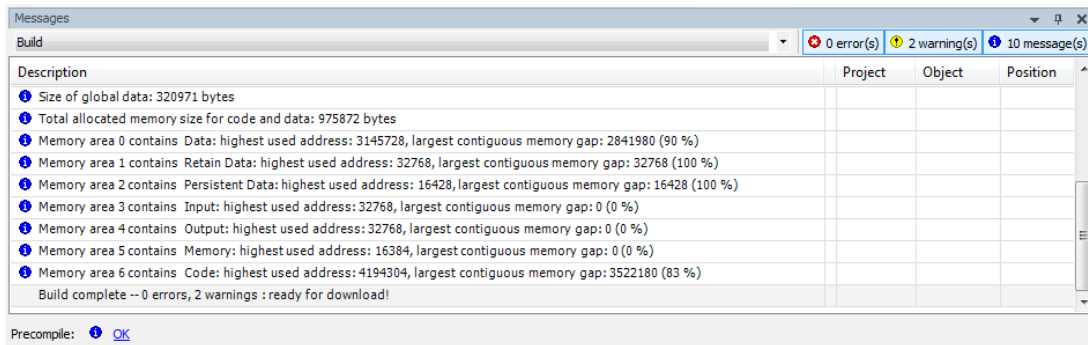


Figure 7-20. Messages Window

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

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 7-20) with exception of the *Precompile* messages, which always are shown in the field below the messages table.

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).

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.

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**.

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.

Watch List View

The Watchlist View provides provides a submenu for opening the available watch lists.

Watch

Symbol: 

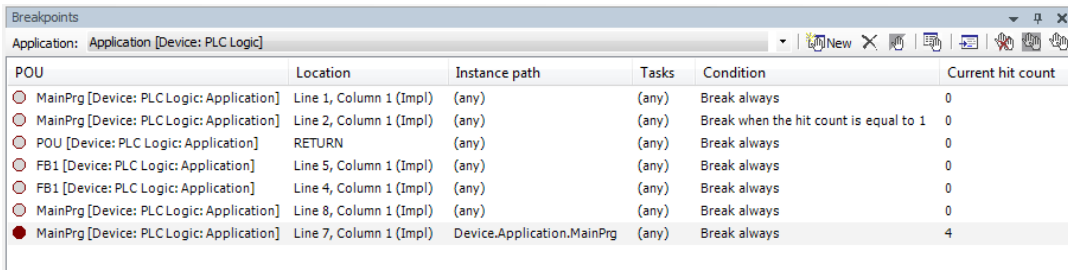
This command opens a submenu with *commands Watch 1, Watch 2, Watch 3, Watch 4 and Watch all Forces*. These are used to open the respective watch list in a view window. *Watch all Forces* is a special watch view for the currently forced values. For further information, see **Debug Menu**.

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**.



POU	Location	Instance path	Tasks	Condition	Current hit count
○ MainPrg [Device: PLC Logic: Application]	Line 1, Column 1 (Impl)	(any)	(any)	Break always	0
○ MainPrg [Device: PLC Logic: Application]	Line 2, Column 1 (Impl)	(any)	(any)	Break when the hit count is equal to 1	0
○ POU [Device: PLC Logic: Application]	RETURN	(any)	(any)	Break always	0
○ FB1 [Device: PLC Logic: Application]	Line 5, Column 1 (Impl)	(any)	(any)	Break always	0
○ FB1 [Device: PLC Logic: Application]	Line 4, Column 1 (Impl)	(any)	(any)	Break always	0
○ MainPrg [Device: PLC Logic: Application]	Line 8, Column 1 (Impl)	(any)	(any)	Break always	0
● MainPrg [Device: PLC Logic: Application]	Line 7, Column 1 (Impl)	Device.Application.MainPrg	(any)	Break always	4

Figure 7-21. Breakpoints

- *Application*: Name of currently active application. Example: “Application [PLC:PlcLogic]”.
- *POU*: Name of POU containing this breakpoint. Example: “MainPrg”.
- *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.MainPrg.FBinst1”.
- *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.

Symbol	Name	Description
	New breakpoint	Opens the “New breakpoint” dialog for defining a new breakpoint. See description of the corresponding command.
	Clean breakpoint	Removes the breakpoint; do not mix up with disabling!
	Enable/Disable breakpoint	Toggles the breakpoint between ‘enabled’  and ‘disabled’  . In case of ‘disabling’ the breakpoint will not be removed from the list, but can be re-enabled.
	Properties	Dialog ‘Breakpoint Properties’ will be opened, where you can modify the breakpoint parameters. The dialog matches the ‘New Breakpoint’ dialog. See the corresponding command for a description.
	Go to source position	The ‘Select Online State’ dialog will open from where you can get to the source position of the breakpoint.
	Clean all breakpoints	Removes all breakpoints. The list will be empty. Do not mix up with disabling!
	Enable all breakpoints	Enables  all currently disabled breakpoints.
	Disable all breakpoints	Disables  all currently enabled breakpoints. The breakpoints remain in the list and can be re-enabled.

Table 7-1. Buttons for Breakpoints

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.

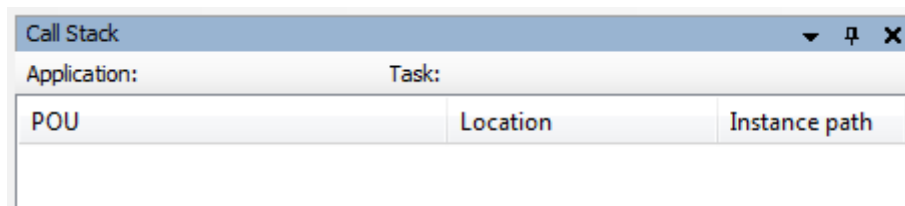
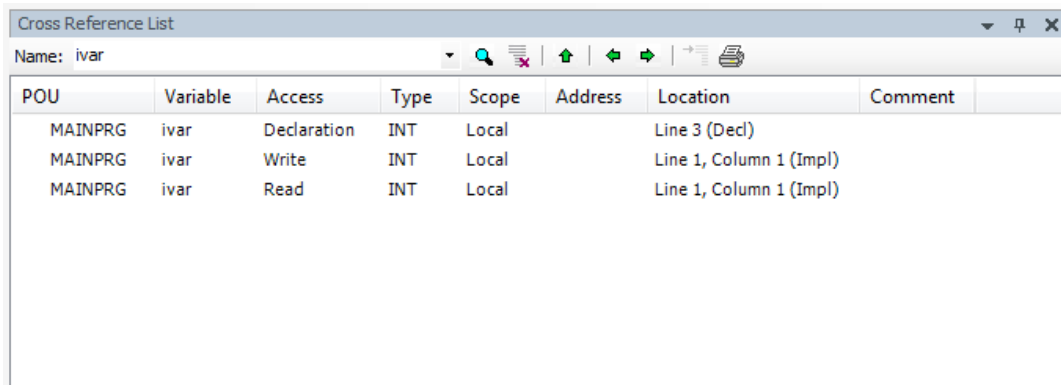


Figure 7-22. Call Stack View

Cross Reference View

This command opens a view window where you can get listed the cross references of a project variable, that is the locations where the variable is used within the project or just within the scope of the same POU.



POU	Variable	Access	Type	Scope	Address	Location	Comment
MAINPRG	ivar	Declaration	INT	Local		Line 3 (Ded)	
MAINPRG	ivar	Write	INT	Local		Line 1, Column 1 (Impl)	
MAINPRG	ivar	Read	INT	Local		Line 1, Column 1 (Impl)	

Figure 7-23. Cross Reference List View

If all cross references within the project should be listed type in the identifier string manually or copy it via copy&paste from any editor window to the *Name* field and then confirm with <ENTER>; or use button .

If only the cross-references within the same POU should be listed: Either select the identifier string within the editor window of that POU and drag it with the mouse into the Cross Reference List view.

If a valid identifier has been entered, the found locations will be listed in the form of a table showing the following information on the variable:

Type	Description
POU	Name of the POU where the variable is used
Variable	Variable name or POU name of the reference
Access	How the variable is used resp. accessed at this location: Declaration / Read / Write / Call
Type	Data type
Scope	Scope of the variable: Global / Local
Address	IEC address if defined
Location	Position of the variable reference within the editor (for example line, network)
Comment	Comment attached at the declaration of the variable

Table 7-2. Information on the Variable

The list can be sorted alphabetically along one of the columns. By a mouse-click on the column title field the sorting will be done in ascending or descending order (toggles with each further mouse-click).

A double-click in a line of the cross reference list the opens the corresponding POU and the respective location of the variable will be highlighted there. This corresponds to using button  (*Show location*) when a list entry is selected.

Button  (*Show previous location*, Shortcut: <SHIFT>+<F4>), will jump to the previous entry in the cross reference list.

Button  (*Show next location*, Shortcut: <F4>) will jump to the next entry in the cross reference list.

Button  (*Go to definition*, Shortcut: <F2>) will jump to the location, where the respective variable is declared. The corresponding declaration editor will be opened and the variable will be highlighted there.

Button  corresponds to command *Browse cross references* and will effect that the current references list is generated to the message window. This might be useful if you want to keep the list available although the automatic display is activated and thus the list in the cross references view might change.

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.

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.

Properties...

Symbol: 

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

Common

It provides information on the object.

- *Full name:* Object name as used in the POU's 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.
- *Description* (only modules): allows to add a description to the modules.

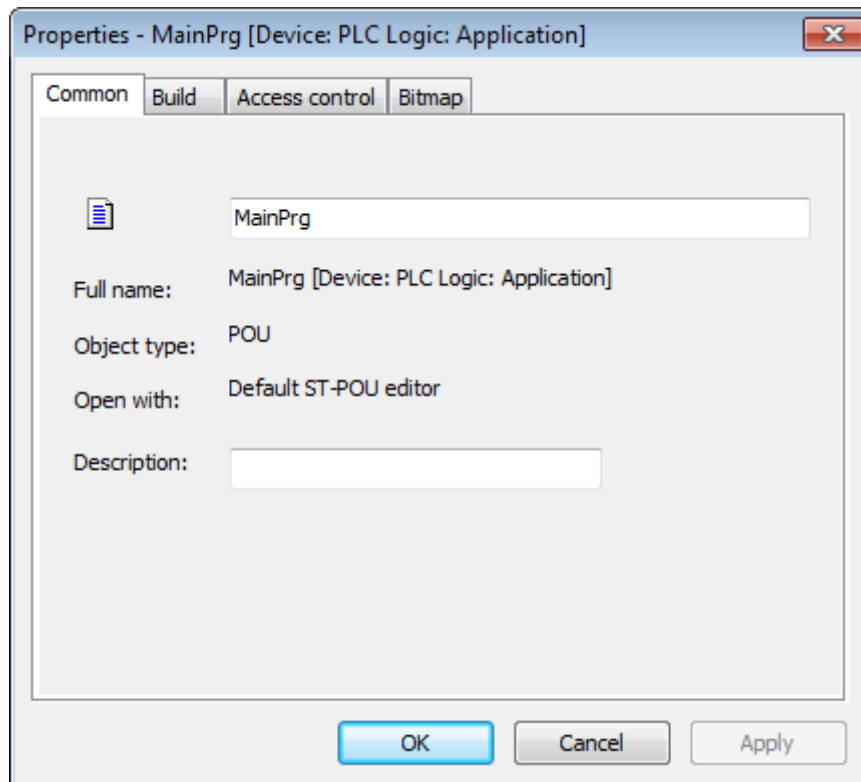


Figure 7-24. Properties Dialog, Common Category

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.
- *Link Always:* The object is marked for the compiler so that it is always included into the compile information. As a result, objects will always be compiled and downloaded to the PLC. This option goes relevant, when the object is located below an application or is referenced using libraries inserted below an application. The selectable variables for the symbol configuration use the compile information as basis too.
- *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.

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

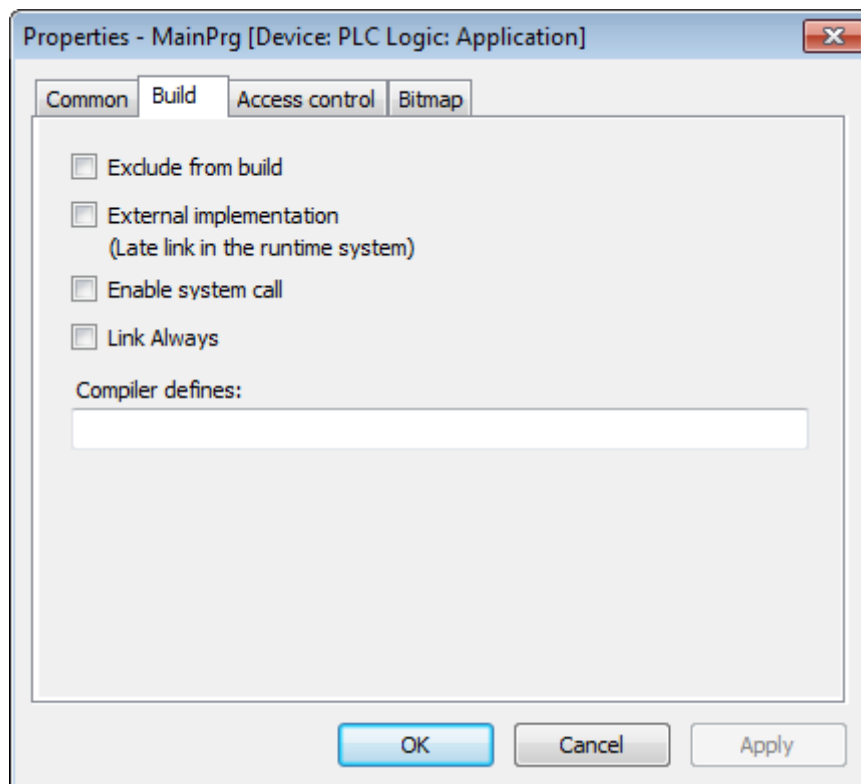


Figure 7-25. Properties Dialog, Category Build

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 *Permissions* dialog, which is available in the *User Management* menu.

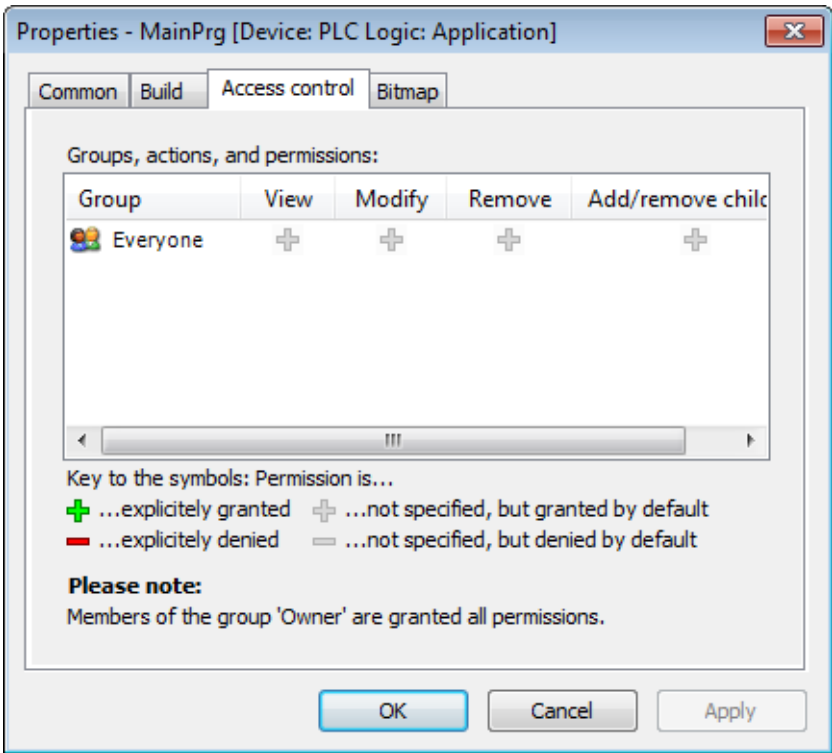


Figure 7-26. Properties Dialog, Category Access Control

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.

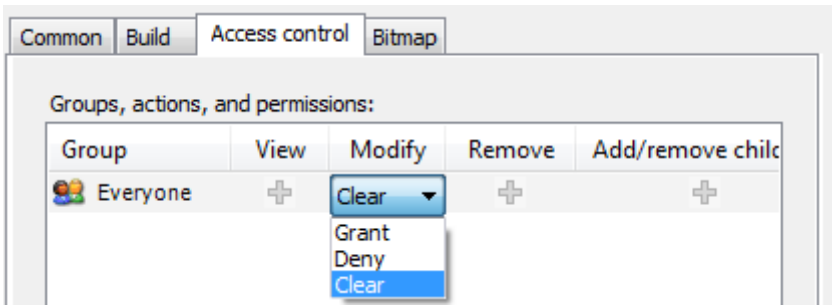


Figure 7-27. Selection List of Rights for action “Modify” for group “Everyone”

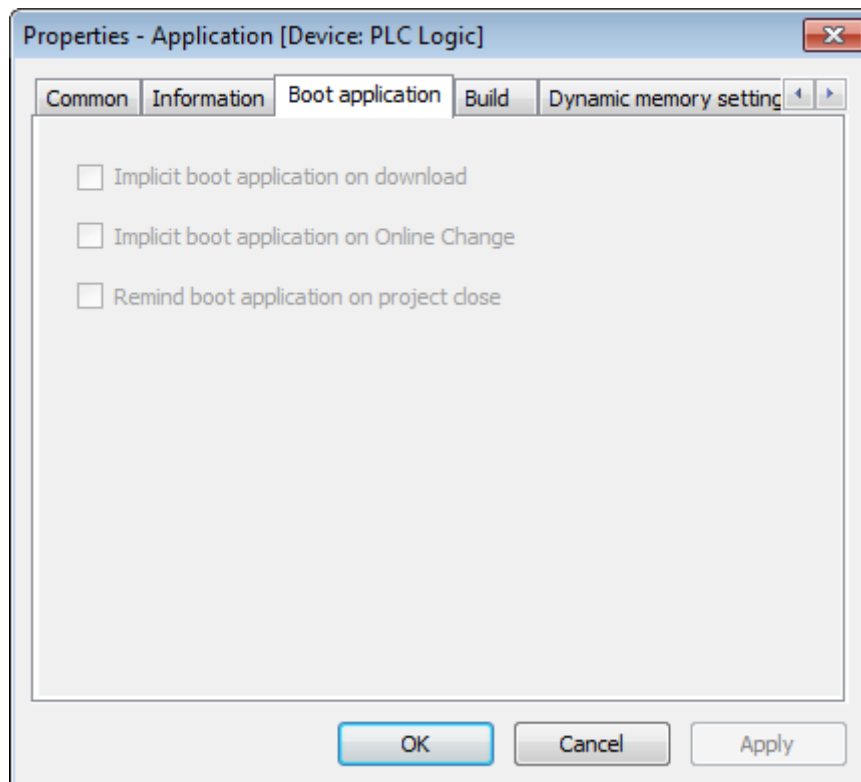
Boot Application

Figure 7-28 Properties, Boot Application Category

It depends on the device whether these settings are available:

- *Implicit boot application on download*: If activated, at a download of the project automatically a boot application will be created.
- *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.

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.

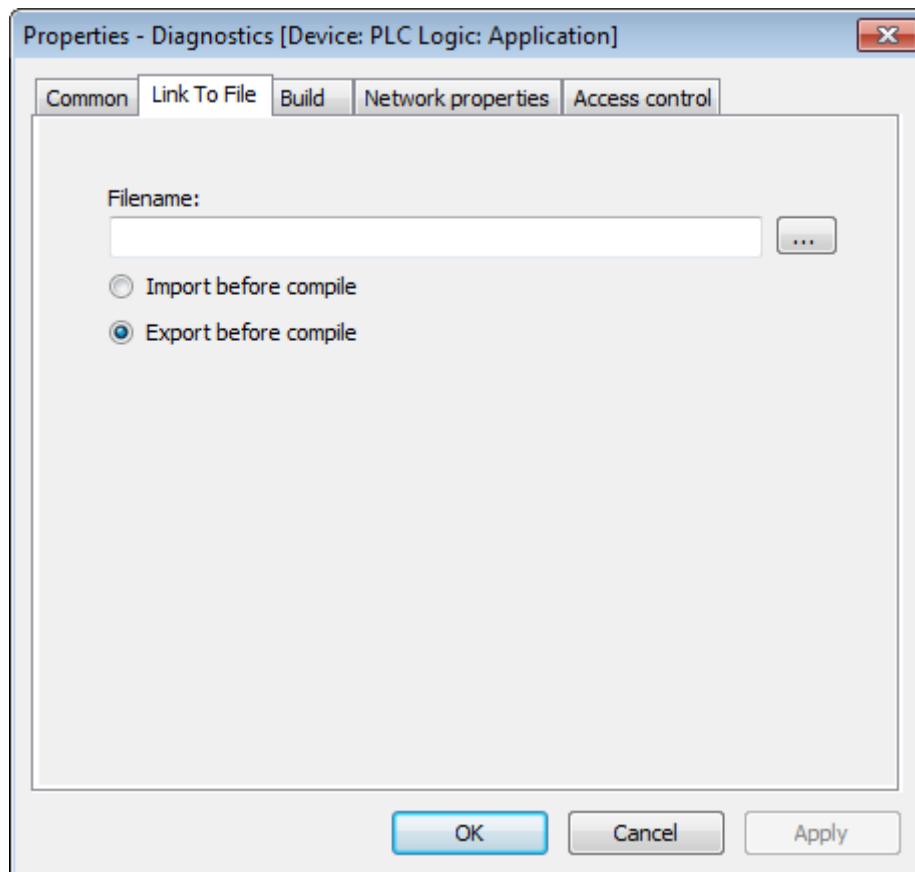


Figure 7-29. Properties Dialog, Category Network Settings

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.

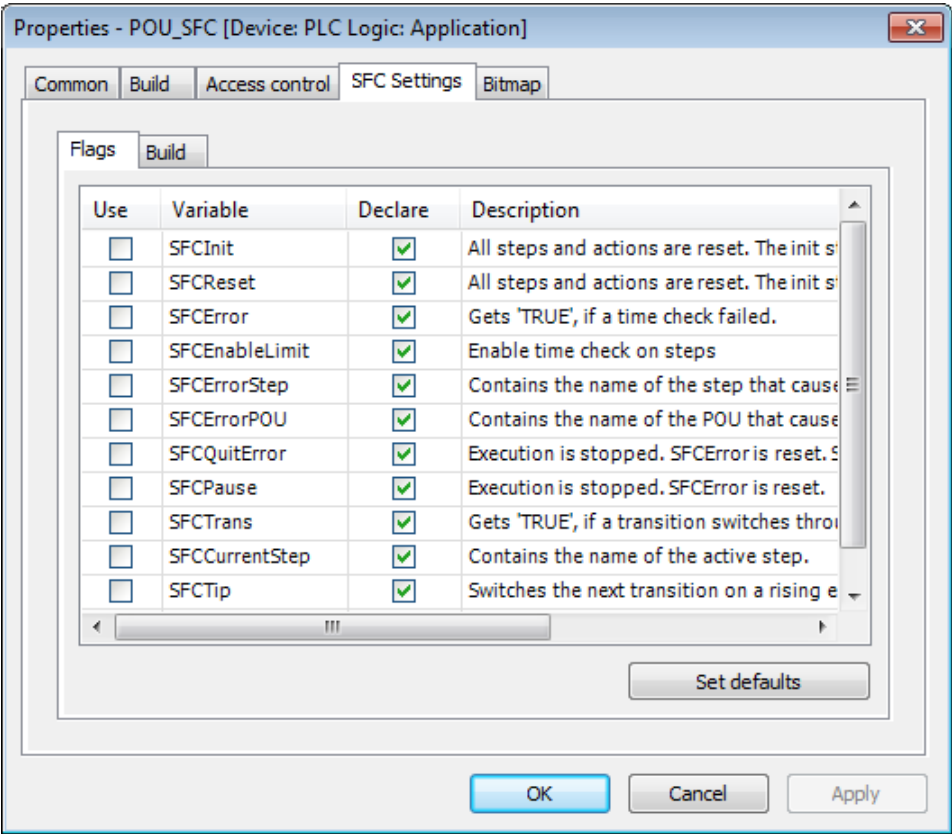


Figure 7-30. Properties Dialog, Category SFC Settings, Flags

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

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**.

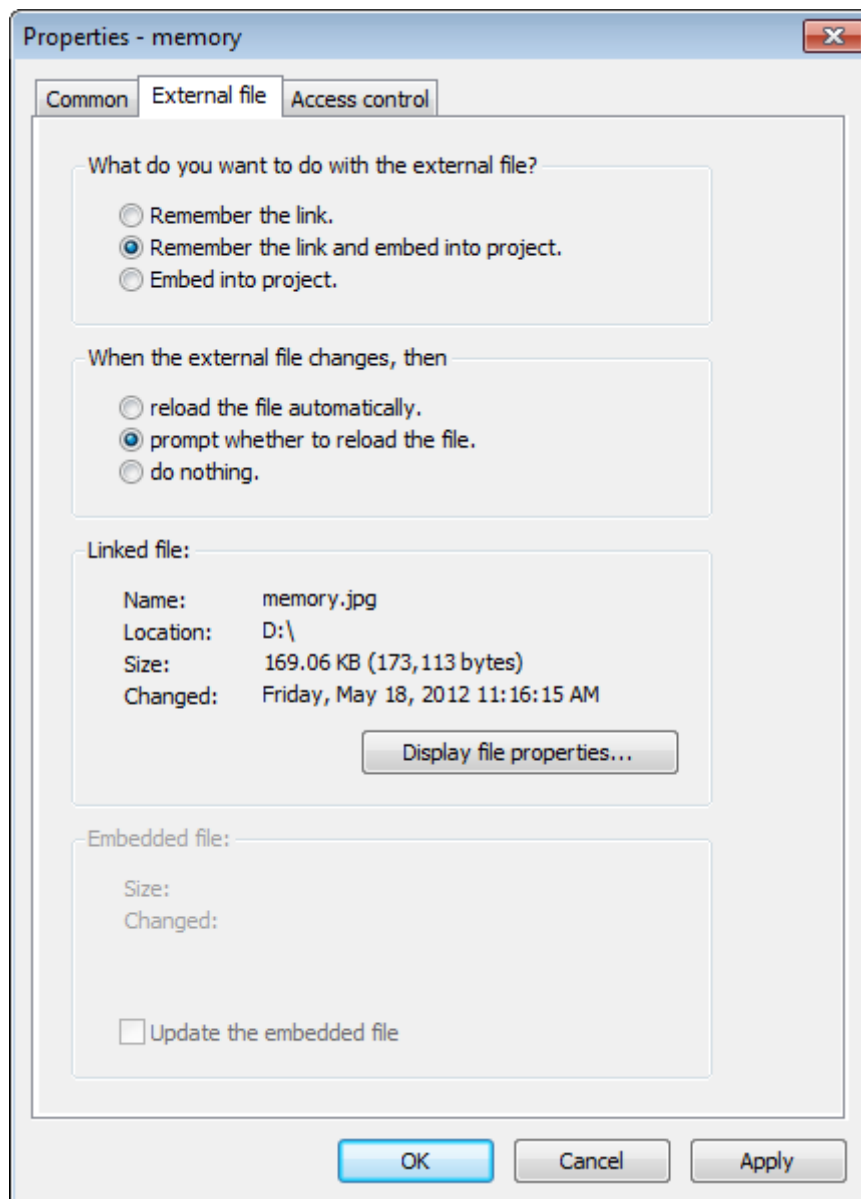


Figure 7-31. Properties Dialog, External File

Project Menu

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

Available command:

- Add Object
- Add Device...
- Add Folder...
- Edit Object
- Edit Object with...
- Set Active Application
- Project Information...
- Project Settings...
- Project Update
- Export to CSV...

- Import from CSV...
- Document...
- Compare...
- User Management

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.

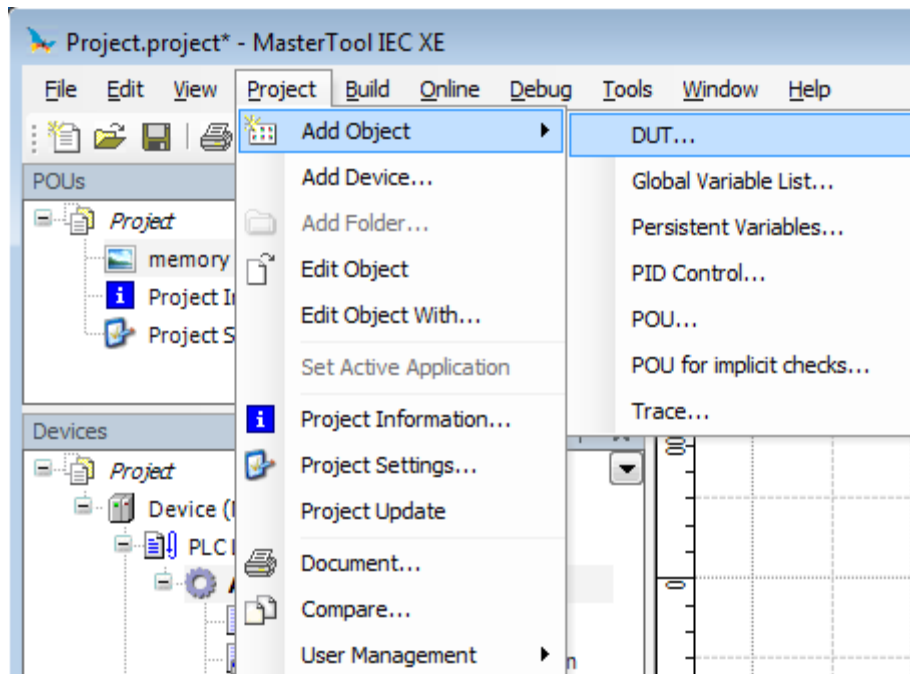


Figure 7-32. “Add Object” Submenu

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.

Add Device

This command is used to add a device object, representing a hardware module, indented below the currently selected object in the device tree.

The command opens the *Add Device* dialog. Depending on the currently selected position in the devices tree, the user will get a list of devices available for insertion. As a precondition, the devices, basing on a device description file, must be installed on your system, which can be done by the *Device Repository* dialog.

In the Name field at top of the dialog define a name with which the device should be entered in the devices tree. This name must be a valid and unique IEC identifier.

By activating the corresponding item you may select which of the actions listed shall be executed: *Append device*, *Insert device*, *Plug device* or *Update device*. Your choice will effect the list of proposals in the sector *Device* as well as the labeling of the button placed at left of the button *Close* for closing the dialog. Currently, only de *Add Device* option is valid.

The scrollable table in the *Add Device* dialog provides a list of devices, which can be inserted at the current position of the devices tree.

The devices are listed with *Name*, *Vendor* and *Version*. By activating the checkbox *Display all versions* (for experts only) all available versions of the particular devices will be displayed as well. The devices might be grouped, that is the table entries (category names) indented in the column name, like for example *Miscellaneous* in the dialog image shown above. To open or close the groups use the preceding plus- and. minus-signs.

By default, initially all available devices will be displayed. If you want to restrict the display on a certain vendor, correspondingly modify the current selection (<All vendors>) in the *Vendor* list above the table.

For the currently selected device additional information - as provided by the device description file - might be displayed below the table: device name, vendor, groups (might be defined for categorization purposes), version, order number and a short description. A device-specific image might be added also.

Depending on your action type chosen you have to confirm your selection by clicking on *Add device*. It is possible to further devices without reopening this dialog. Thereby you may select the appropriate node by a click in the device tree.

Having finished you may quit the dialog by pressing *Close*.

This command is also available in the context menu of certain devices in the device tree.

NOTE: A device can also be added to the devices tree by the *Add Object* dialog. Using the *Add Device* dialog provides the advantage of getting displayed additional information on the device.

Add Folder

Symbol: 

This command will be available in the *Project* menu and also in the context menu if you are working in the POU's 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.

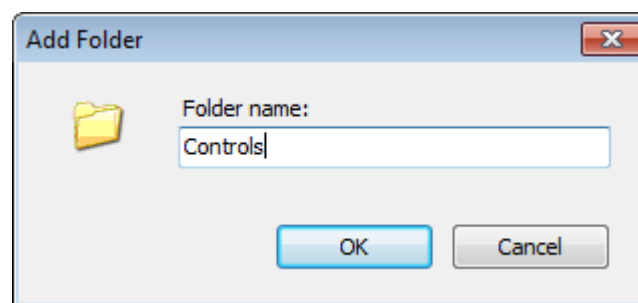


Figure 7-33. Add Folder

After confirming with *OK* the folder will be inserted in the tree preceded by the folder icon .

If the folder has been created below an existing tree entry, click on the plus-sign, which will now appear before that entry to make visible the new folder.

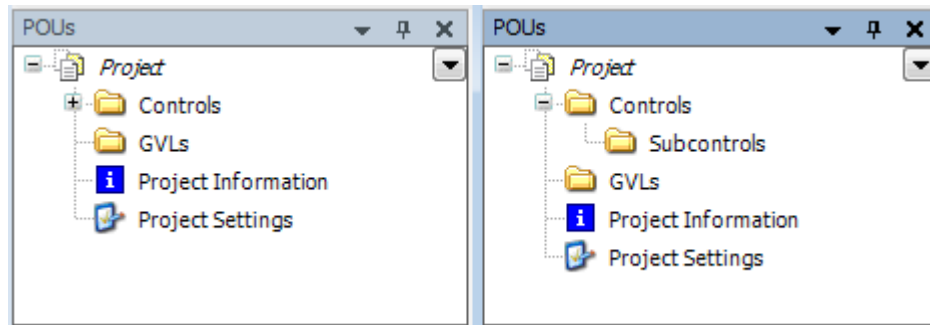


Figure 7-34. Folders in POUs View

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.

Edit Object with

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

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*.

File

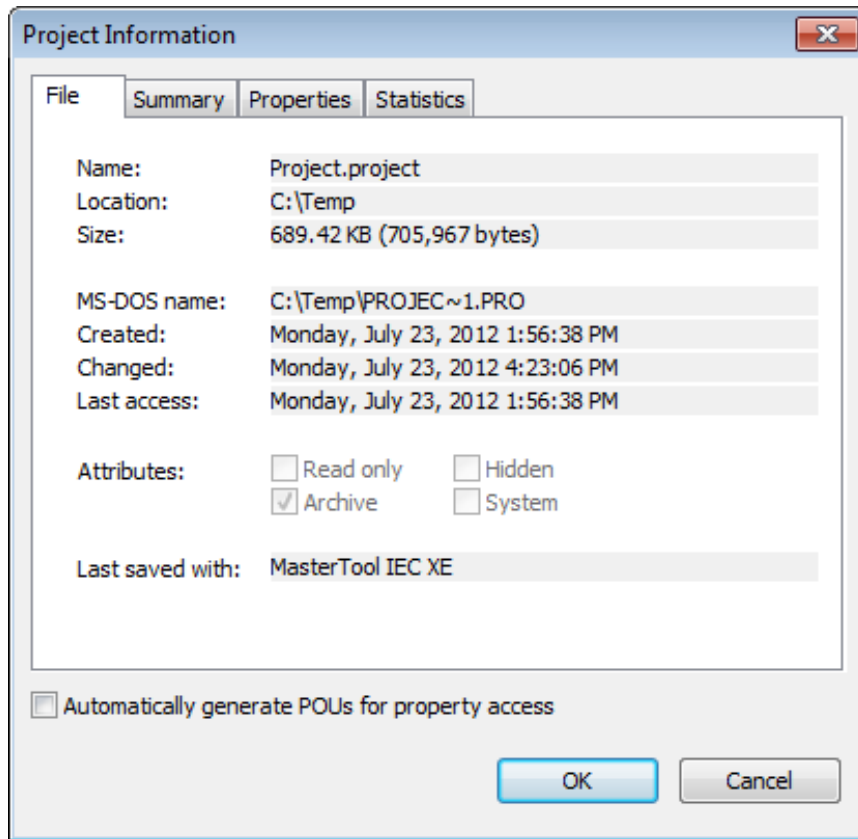


Figure 7-35. Project Information Dialog, File

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*).

Summary

The screenshot shows the 'Project Information' dialog box with the 'Summary' tab selected. The dialog contains several input fields: 'Company:', 'Title:', 'Version:', 'Library Categories:', 'Default namespace:', 'Author:', and 'Description:'. The labels for 'Company:', 'Title:', and 'Version:' are in bold. To the right of the 'Version:' field is a checkbox labeled 'Released'. To the right of the 'Library Categories:' field is a button with three dots. At the bottom of the dialog, there is a checkbox labeled 'Automatically generate POUs for property access' and two buttons: 'OK' and 'Cancel'. A note at the bottom of the dialog states: 'The fields in bold letters are used to identify a library.'

Figure 7-36. Project Information Dialog, Summary

Here you can optionally add some information on the project file like a *Title*, a *Version*, the *Default Namespace*, the *Author*, the *Company name* and any *Description* text. 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.

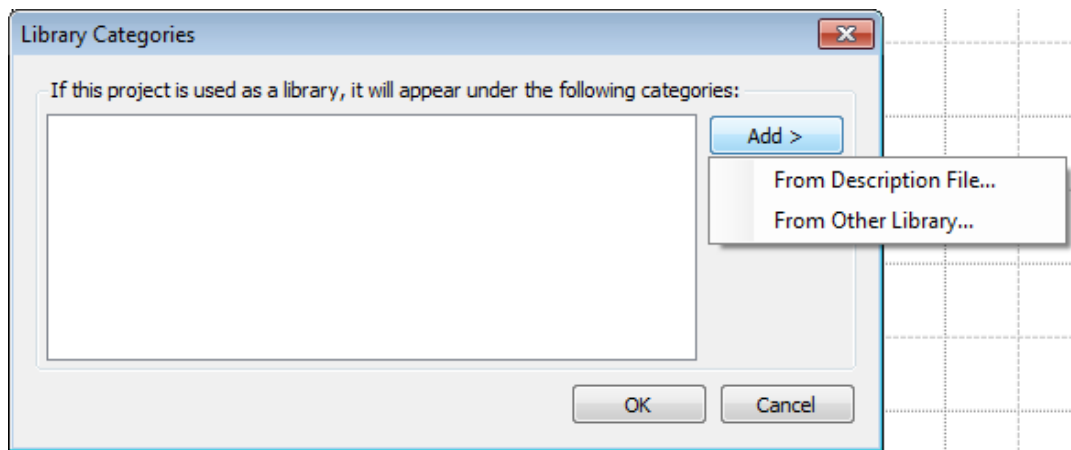


Figure 7-37. Library Categories Dialog

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.

Properties

Project Information

File Summary **Properties** Statistics

Key: Add

Type: Text Modify

Value: Remove

Properties:

Key	Value	Type

☐ Automatically generate POUs for property access

OK Cancel

Figure 7-38. 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 POUs for property access: 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 (GetTextProperty, GetBooleanProperty, GetNumberProperty, 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.

Example: The following property is defined in the Properties tab: Key = nProp1, Type = Number, Value = 333. To get the value in the application program, call function GetNumberProperty, for example showprop:=GetNumberProperty("nProp1"); showprop must be declared as type DINT in this case.

Statistics

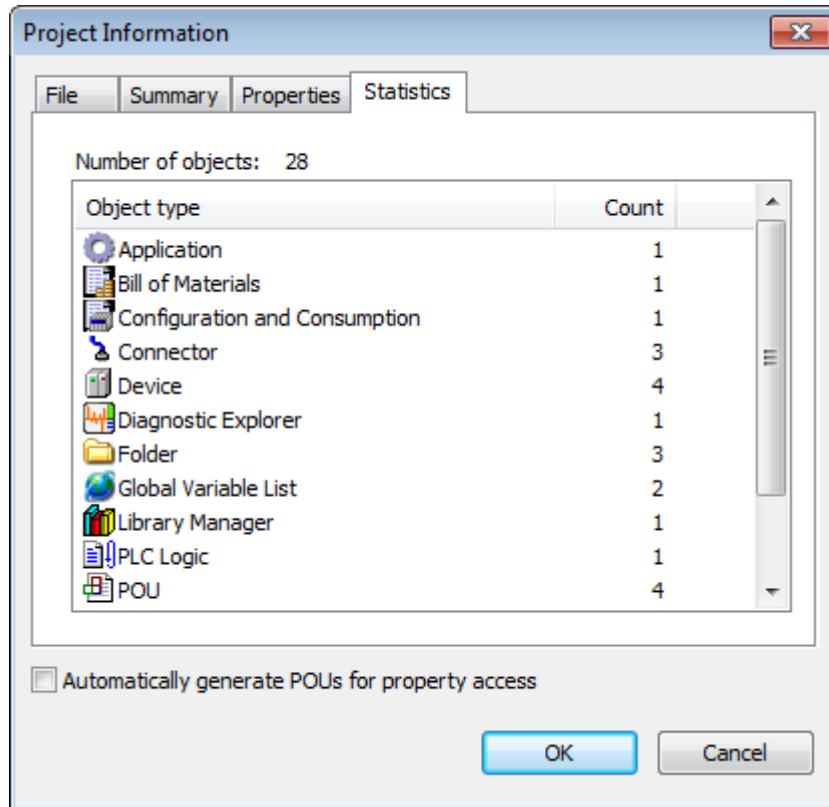


Figure 7-39. Project Information Dialog, Statistics

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.

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.

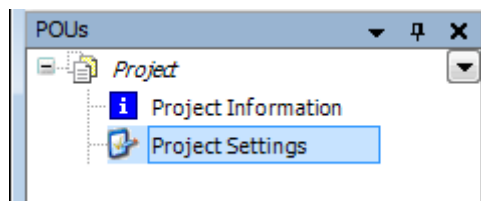


Figure 7-40. Project Settings Object in POUs Window

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

Page Setup

These commands are related to the dialogs for configuring the layout of a printout page. The layout defined here is valid for printouts via the *Print...* and *Document...* commands.

Source Download

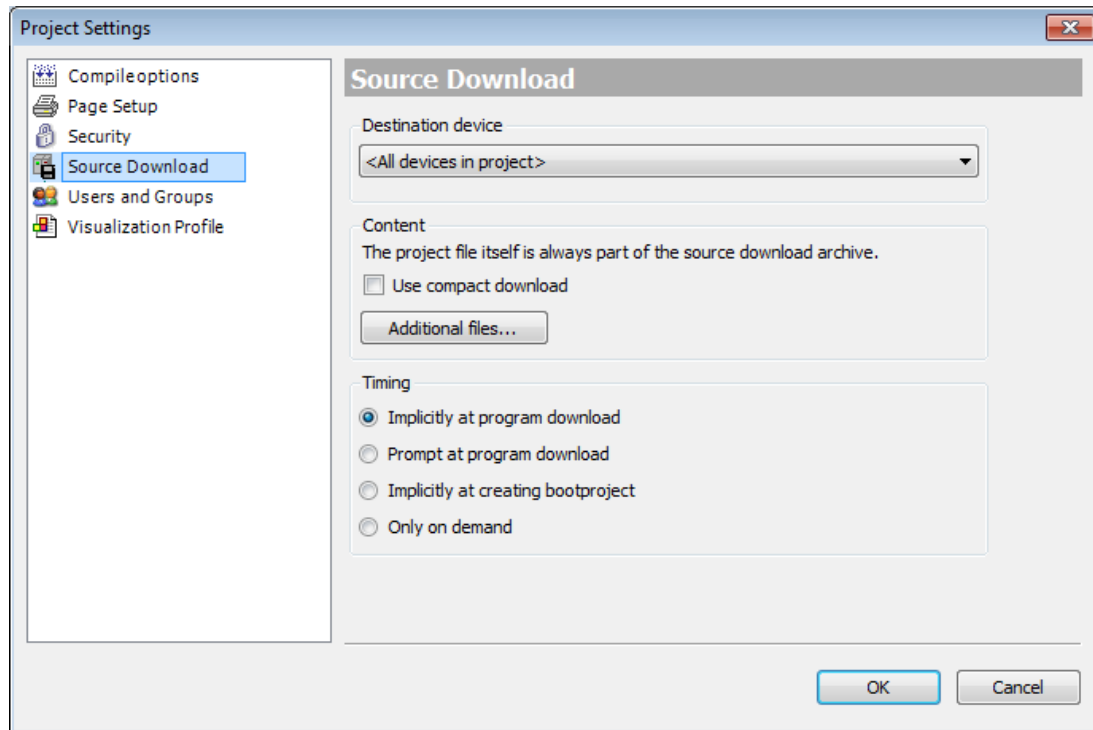


Figure 7-41. Project Settings Dialog, Category Source Download

The following settings are valid for a source download that is transferring the project file to the PLC:

- *Destination device*: By default no device is predefined here. However, you can select one from the selection list, which offers all programmable devices defined in the project.
- *Content*: Predefines the selection of project-belonging files to be stored in the archive file for source download.
- *Timing*: Predefines when a source download will be done.
 - *Implicitly at program download*: The source download is executed at each program download without any user interaction.
 - *Prompt at program download*: At each program download the user will be asked whether also a source download should be done.
 - *Implicitly at creating bootproject*: The source download is executed at each boot project creation without any user interaction.
 - *Only on demand*: The source download has to be initiated by command *Source Download*.

Compile Options

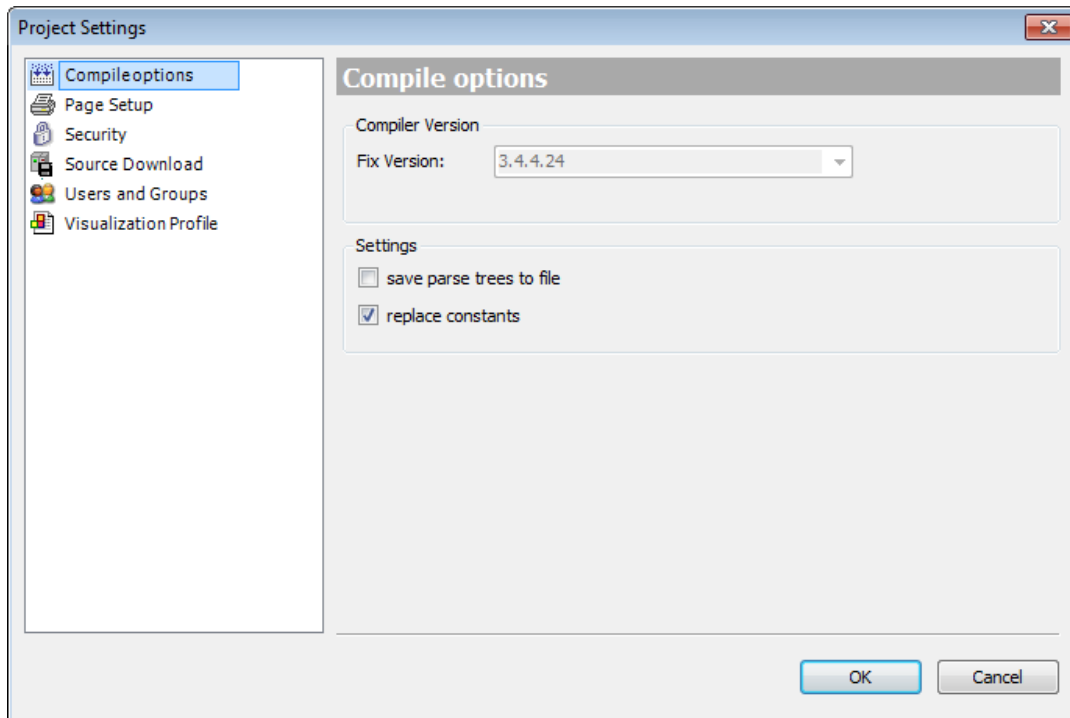


Figure 7-42. Project Settings Dialog, Category Compile Options

Compiler Version: Here the user can define which compiler version should be used when the project gets compiled for example by one of the build commands or during download.

NOTE: Whether the data type LREAL will be treated as 64-bit type or converted to REAL is determined by the particular target device.

Settings:

- *Save parse trees to file:* In case of large projects or low-performance systems, it might be reasonable to activate this option. It helps to avoid out-of-memory problems by storing the precompile (language model) information of the project in a temporary file and not in the main memory. However, this will increase the project loading and compilation time.
- *Replace constants:* Per default this option is activated, which means: For each constant of scalar type (thus not for strings, ARRAYS, structures) directly its value gets loaded. In online mode the constants are indicated by an  icon preceding the value in the declaration or watch view. In this case any accessing of constants e.g. via an ADR operator or forcing and writing are not possible. If the option is deactivated, the constant can be accessed, however this will mean a longer processing time.

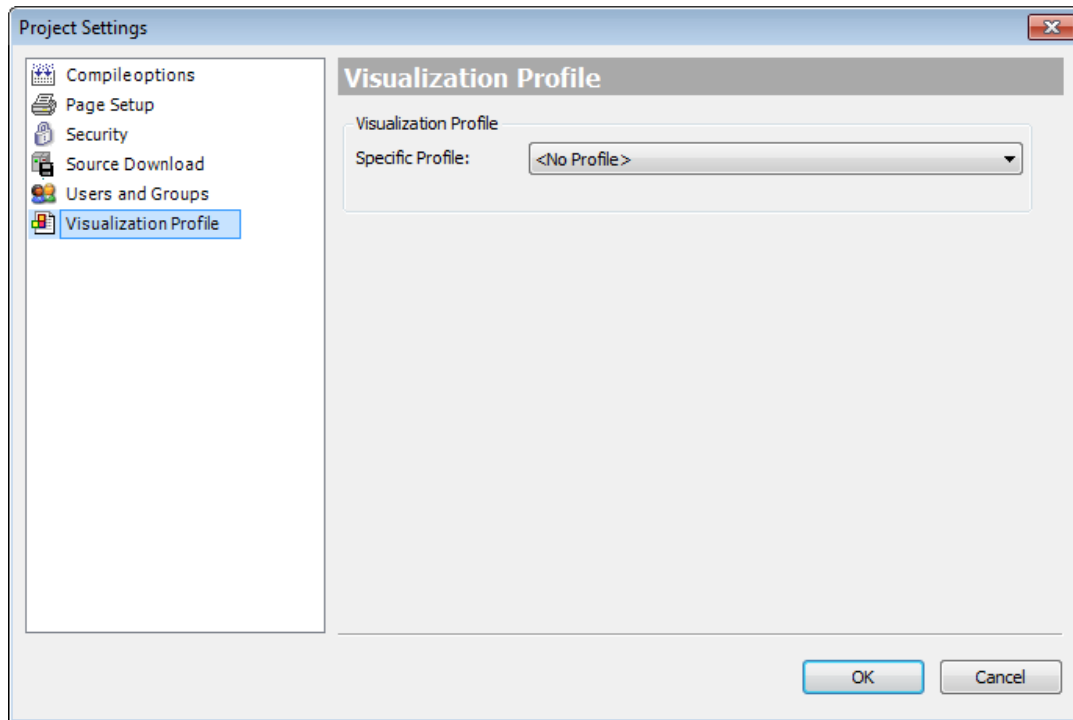
Visualization Profile

Figure 7-43. Project Settings Dialog, Category Visualization Profile

Define here a visualization profiles to be used when the project gets opened. Currently there is no visualization profile available to Nexto Series CPUs:

Security

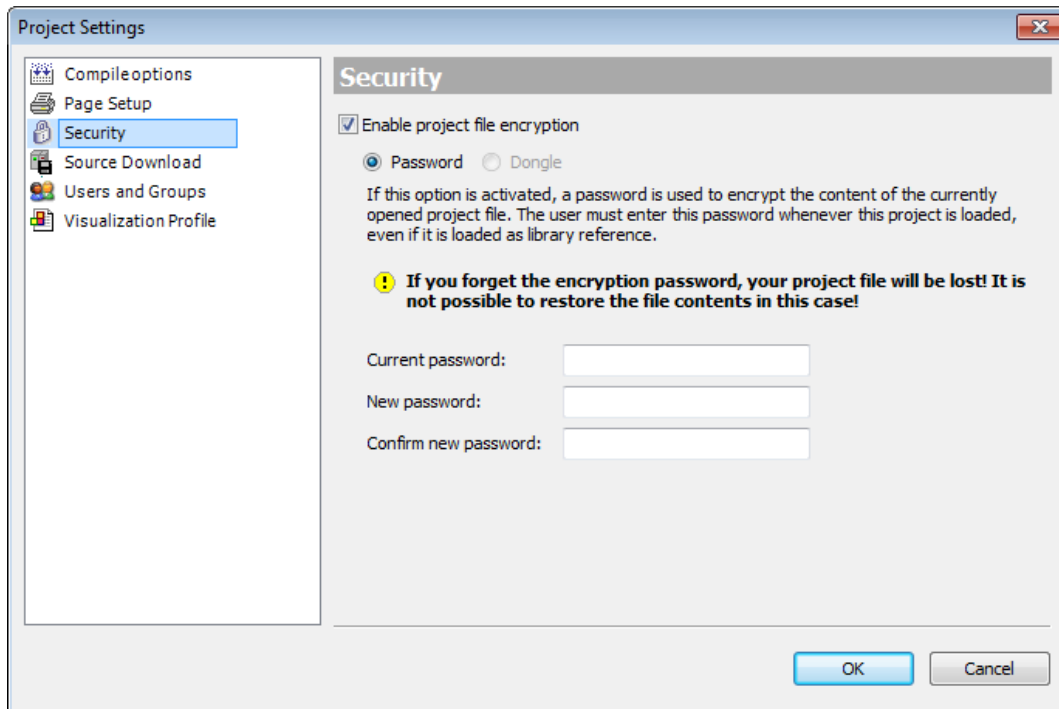


Figure 7-44. Project Settings Dialog, Category Security, Password

This dialog allows the use of a project password. Having activated the option *Enable project file encryption* the password has to be entered in the edit fields *New Password* and *Confirm password* (in order to detect type errors). If the checkbox is still activated while saving the project, this password will be required for a reload of the project, even if it will be loaded as a library reference. The *Encryption Password* dialog will open in this case.

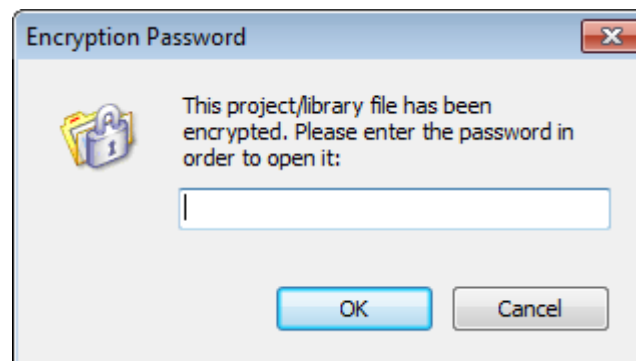


Figure 7-45. Encryption Password

NOTE: If you do not remember the encryption password, the project will be lost! File contents cannot be restored in this case.

The project settings dialog may be reused for modifying the password. Therefore, in edit field *Current password*, you have to enter the actual password first, before you are allowed to change it by entering your new suggestion twice in the edit fields *New Password* and *Confirm password*.

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*

Users

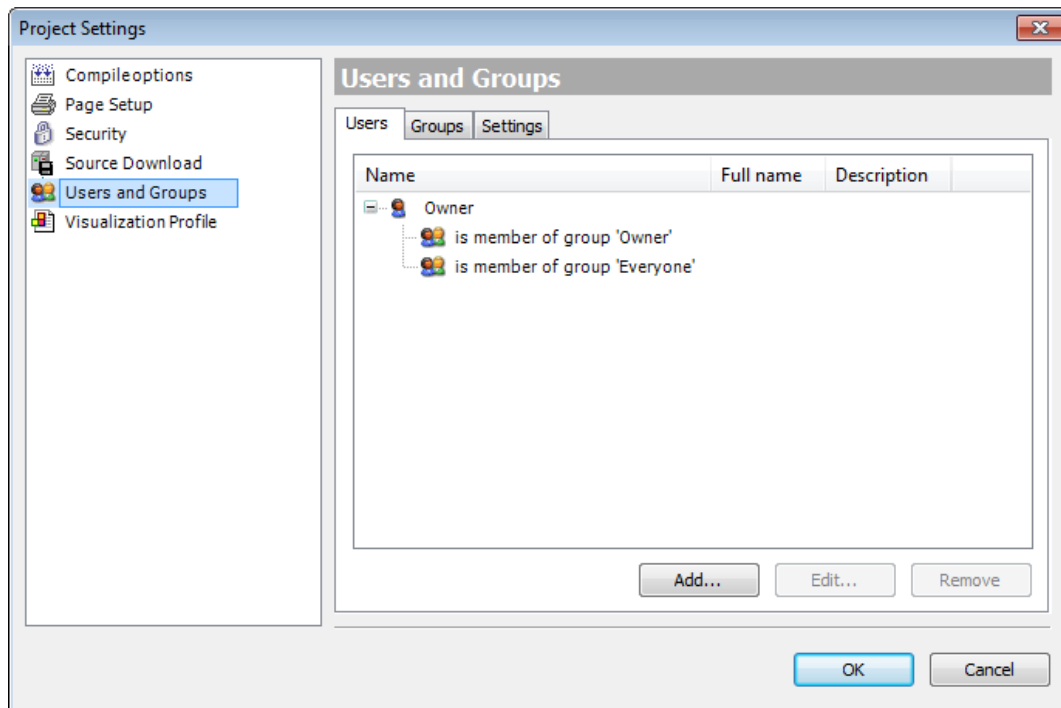


Figure 7-46. Project Settings, Users dialog

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.

Add User

Account properties:

Logon name: user1

Full name: user1

Description: User Boss

Old password:

Password: *****

Confirm password: *****

Active: ☒

Memberships:

☒ Owner

This user is also member of the 'Everyone' group.

OK Cancel

Figure 7-47. 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.

Groups

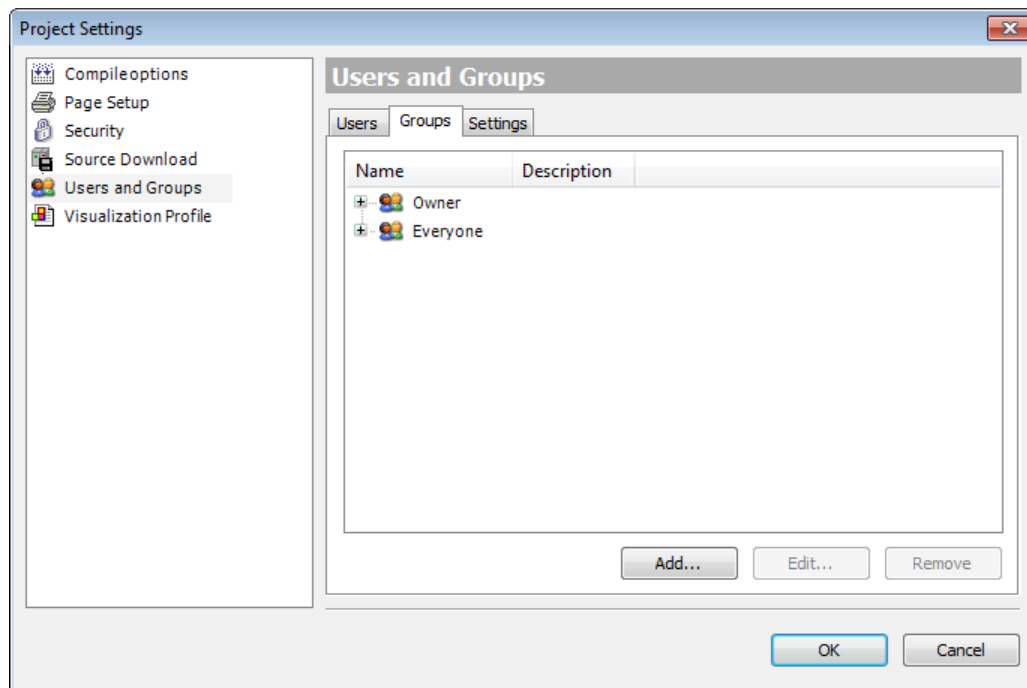


Figure 7-48. Project Settings, Groups Dialog

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.

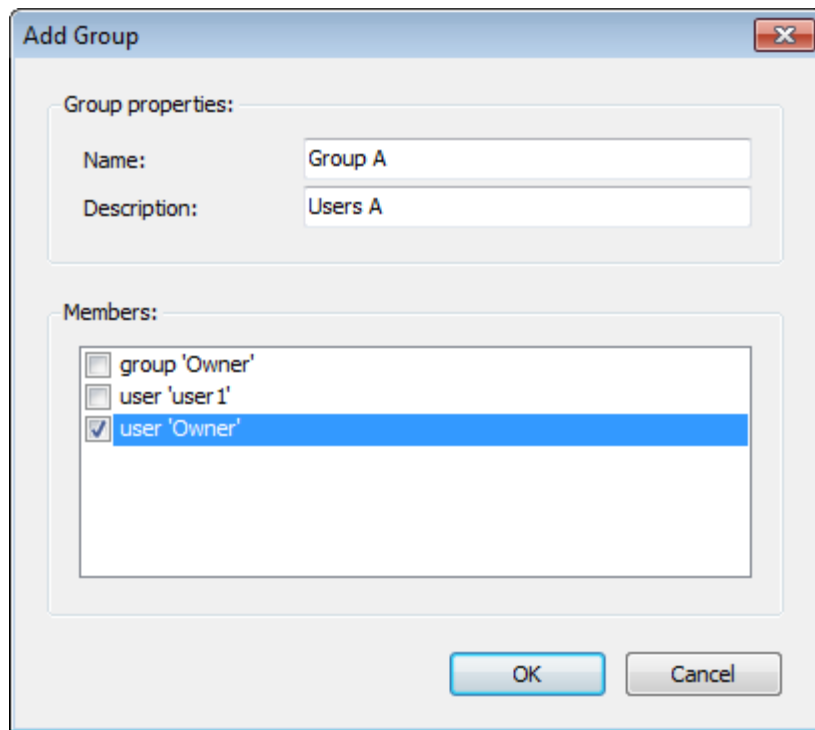


Figure 7-49. Add Group Dialog

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 button *Edit* to open the *Edit Group* dialog. The entry fields are the same as in the *Add Group* dialog (Figure 7-49). 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*.

Settings

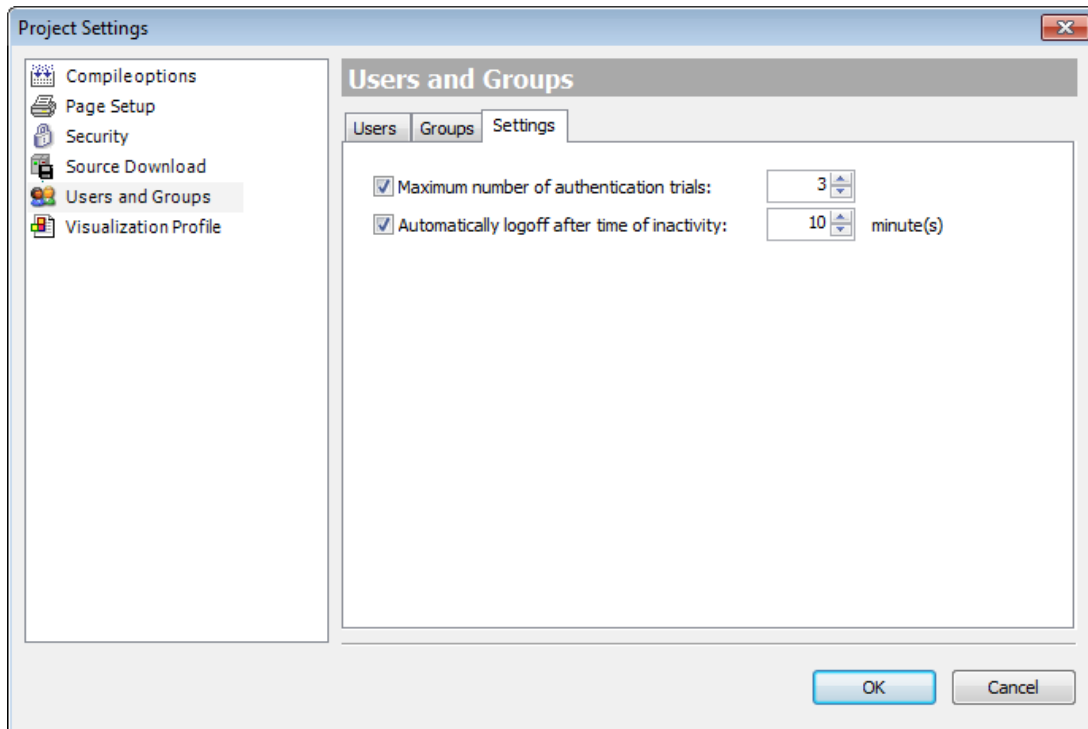


Figure 7-50. Project Settings, Users and Groups Settings Dialog

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.

Project Update

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

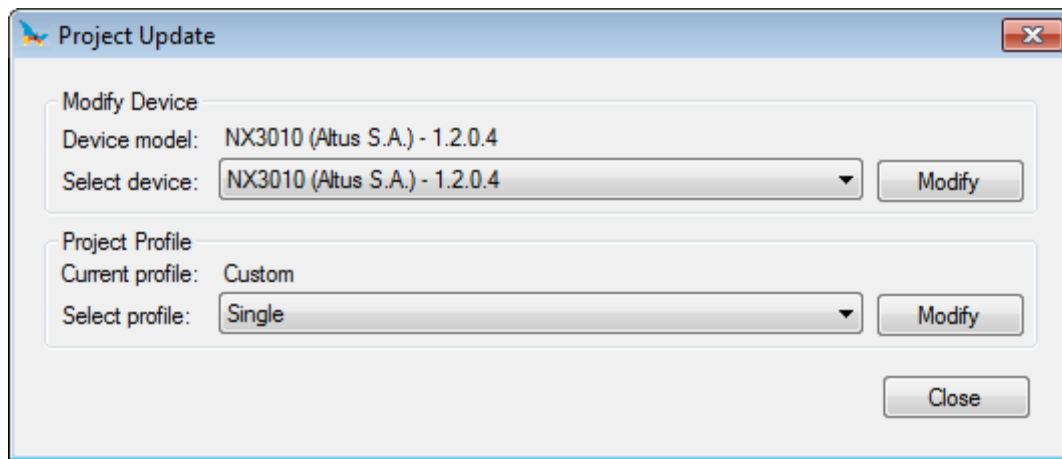


Figure 7-51. Project Update

Modify Device

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

- *Device model*: displays the model, the vendor and the version of the device present in the project.
- *Select device*: displays all the available options for modification or updating of models and versions of CPUs.
- *Modify*: performs the modification or update selected by the user, thus changing the model or the versions of devices currently used in the project.

During the change or update process the user will be prompted to keep the parameters defined for each device. Choosing "Yes" all parameters are kept as the original project. However, if there is a change in the format of these device parameters it will not be applied. See chapter compatibility between versions of the documentation to see if was no format changes in the parameters. Choosing "No" the parameters format of devices will be updated as amended. However, the parameters will be changed to the configurator default values.

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 NX3010 CPU with its default name kept, after the CPU modified to NX3020 will keep your name as "NX3010", 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.

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 MasterTool. 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 NX3010 UCP model the upgrade should be made to the most current version of this same model.

Before starting an update, it is important to check the documentation of the modules used in the project. If there is any incompatibility between the parameters from one version to another, the behavior will be different. If this is not the case, choose “Yes” for all the updated devices, as shown on Figure 7-52. In case a new parameter is added or existent parameters are modified, it is not possible to keep the same configurations. Then it is necessary to choose “No” and add the module to the default parameters of the project.

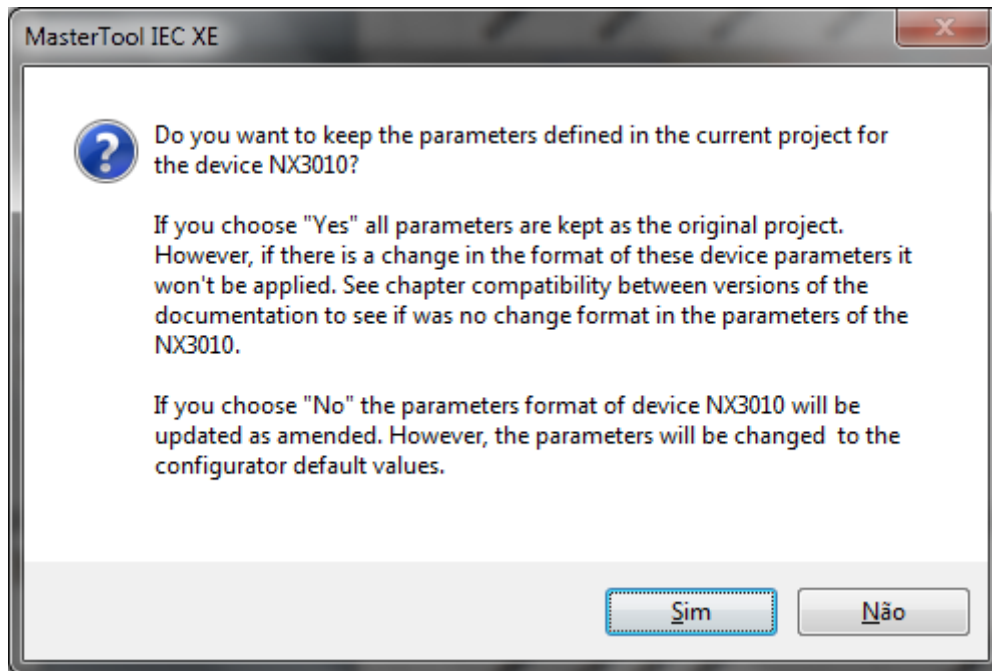


Figure 7-52. Device Update Message

NOTE: Changes in modules parameters are described in each module documentation in the chapter about compatibility.

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.

Export to CSV

This command allows you to export all devices instantiated in the project or a particular device to an external file in CSV format. The export of all devices occurs via *Export to CSV...* in the *Project* menu. To export a particular device, you must access the context menu of this device and choose the option *Export to CSV...*

By choosing the *Export to CSV...* option the *Save As* dialog box appears. It is possible to define the name of the export file and the location where it will be saved. “Devices sons” of a device selected for export are also exported. After you click *Save*, the export starts. The code below shows the contents of a file generated by the export command:


```
#ModbusClient;Parent;Connector;Protocol;TaskCycle
"MODBUS_Symbol_Client";"NX3030";"NET 1";Tcp;100
#ModbusClientDevice;Parent;SlaveAddress;IP;TcpPort;MaximumSimultaneousRequest;CommunicationTimeOut;Mode;InactiveTime
"MODBUS_Device";"MODBUS_Symbol_Client";"1";"[192, 168, 19, 135]";502;1;3000;CloseByTime;10
#ModbusClientDeviceMapping;VariableName;DataType;StartAddress;Quality;AbsoluteAddress
"MODBUS_Device";"GVL.var_int";HoldingRegisterWrite;1;"GVL.var_quality";40000
#ModbusClientDeviceRequest;Function;Polling;ReadDataStart;ReadDataSize;WriteDataStart;WriteDataSize;Diagnostic;Disable
"MODBUS_Device";FC16;500;1;1;1;1;"GVL.var_diagnostic";"GVL.var_disable"
```

Figure 7-53. Exported Device Example

In the export file, lines beginning with the character (#) describe the name of the parameters of a device, mapping, or request. The subsequent lines contain the values for these parameters. In this file, the information is separated by semi-colon (;). The export file can be imported into an existing project via the import command.

NOTE: The devices, which cannot be exported, have the *Export to CSV...* option disabled. In the same way, these devices will not be exported if it is used the option available in the *Project* menu.

NOTE: This feature is available from the version 1.40 of MasterTool IEC XE to later.

Import from CSV

This command allows you to import one or more devices described in the command generated by the export file. To import all the devices described in the file, you must use *Import from CSV...* available on the *Project* menu. Partial import occurs when you want to import only a device described in the file. For this, the device to be imported must be instantiated in the project and its name must match the name of the device described in the file. Thus, the partial import may occur via *CSV Import from CSV...* on the context menu of the target device.

By choosing *Import from CSV...* option, the *Open* dialog box appears. It is possible to choose the file generated by the export command. After selecting the file and clicking *Open*, the import will be started and the devices described in the file will be added to the current project. The import will be undone if errors occur during the process and appropriate messages will be displayed describing the error.

The import command operates under existing projects. Thus, some decisions are made during the import when the devices are (or not) previously instantiated in the project. These decisions are described below:

- If a device described in the file does not exist in the project, it will be created, unless it is a module. In this case, it must be previously existent.
- If a device already exists in the project, its settings will be replaced by the configuration described in the file.
- If an instantiated device in the project is not described in the file, it will be removed from the project unless it is a module. In this case, it will be kept in the project with the same settings.

NOTE: The Import command does not create new projects. Thus, the option *Import from CSV...* on *Project* menu is enabled only if the project is opened.

NOTE: In the case of import process triggered by the selection of a device, there is no error return if there are no data in the import file to be imported into the device.

NOTE: This feature is available from the version 1.40 of MasterTool IEC XE to later.

Document

This command opens the *Document Project* window, where you can configure and print a documentation for the project.

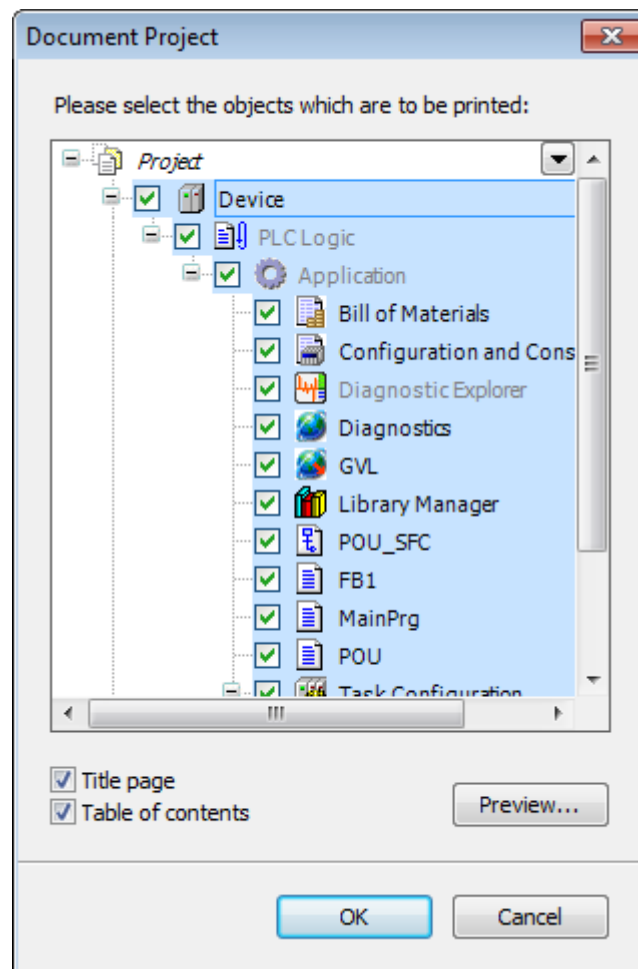


Figure 7-54. Document Project

The project objects in POU 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.

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.

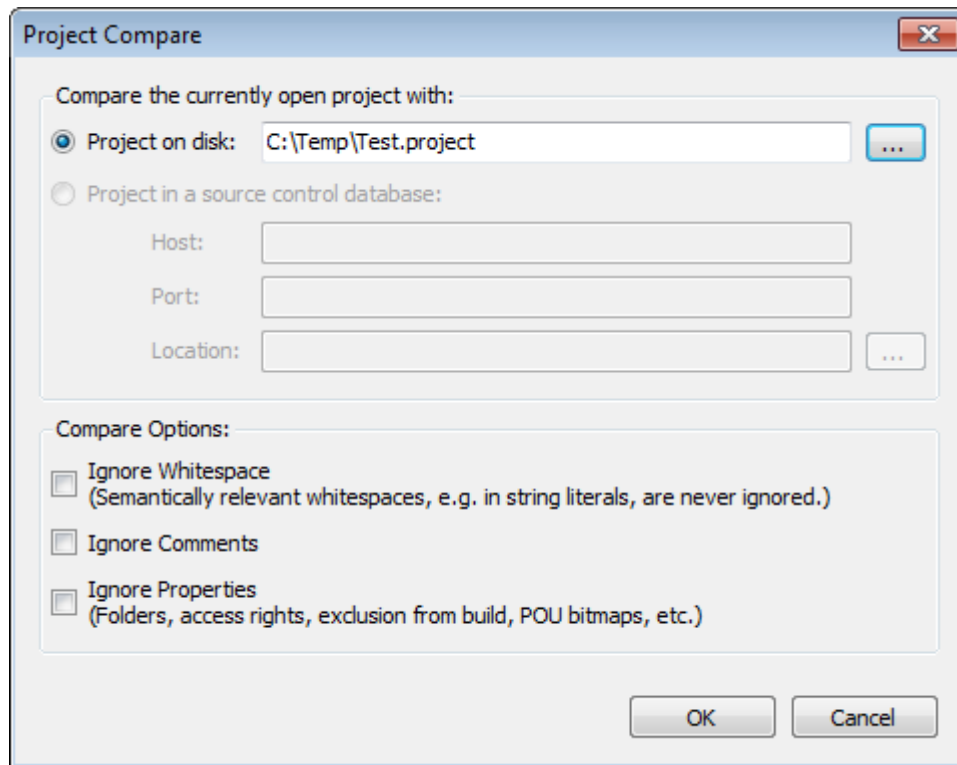


Figure 7-55. Project Compare Dialog

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.
- *Project in a source control database*: where you first have to specify the host name as well as the number of the port. Afterwards you may insert the path of the designated project in the text field or select it via 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.

Survey of Comparison Result by means of Marked Device Trees

The result of the comparison is represented in a new window entitled *Project Compare - Differences*.

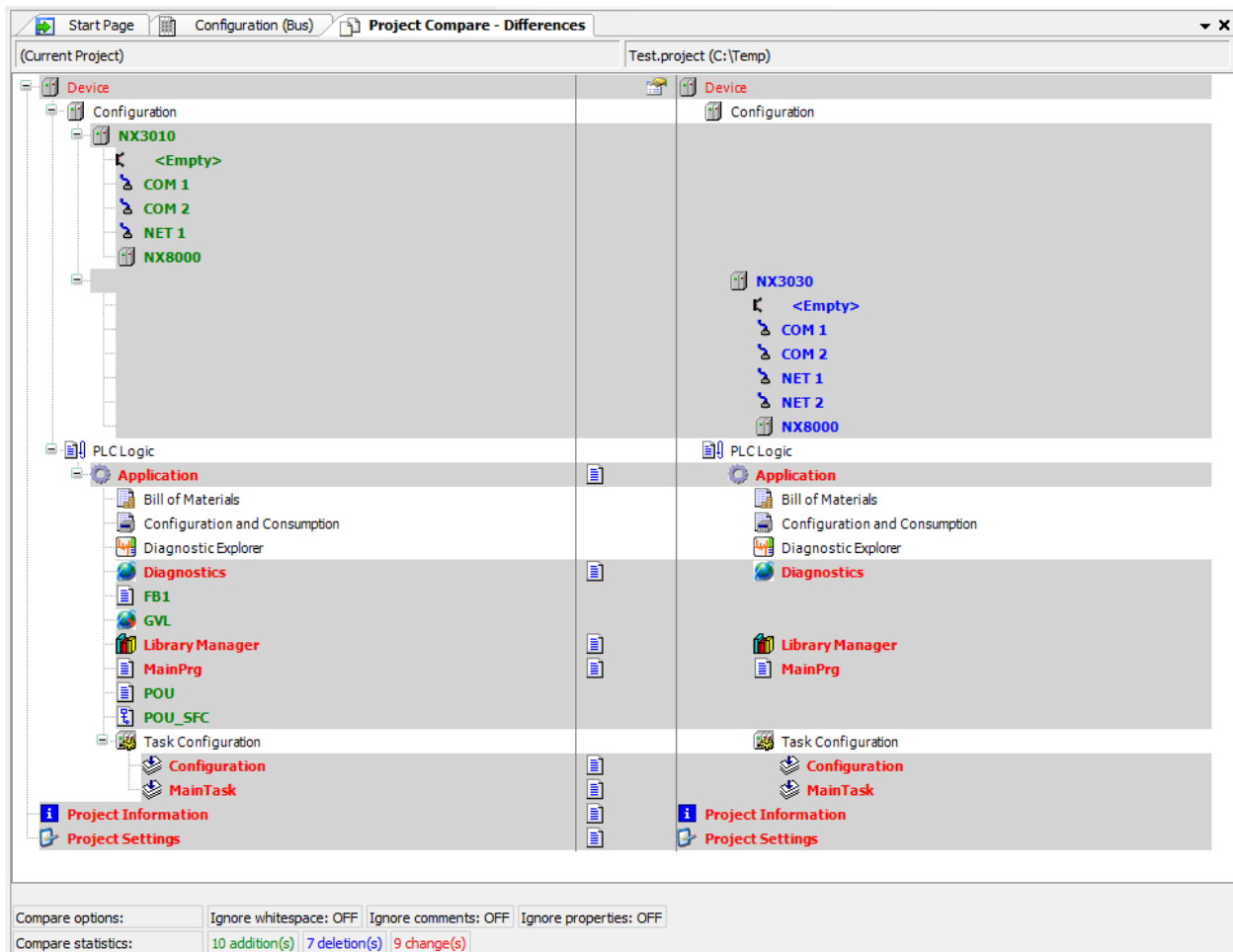


Figure 7-56. Comparison Result

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 (indicated by  subsequent to the unit name) access rights (indicated by  subsequent to the unit name) or added folders (indicated by  subsequent to the unit name). In the last case, the corresponding path of the unit concerned is displayed straight below the sub-window as soon as you place the mouse pointer on its name in the actual project or the reference project respectively.

In addition, each line containing a colored entry is highlighted in grey.

Below the sub-window, the user find further lines displaying the options set for the comparing as well as a short statistic showing the number of additions, deletions and changes.

You may achieve detailed information on discrepancies of units marked in bold red by a double click on their name. After that, the survey will be superposed by a detailed view (Figure 7-57). Press in the <BACKSPACE> key to return to the survey.

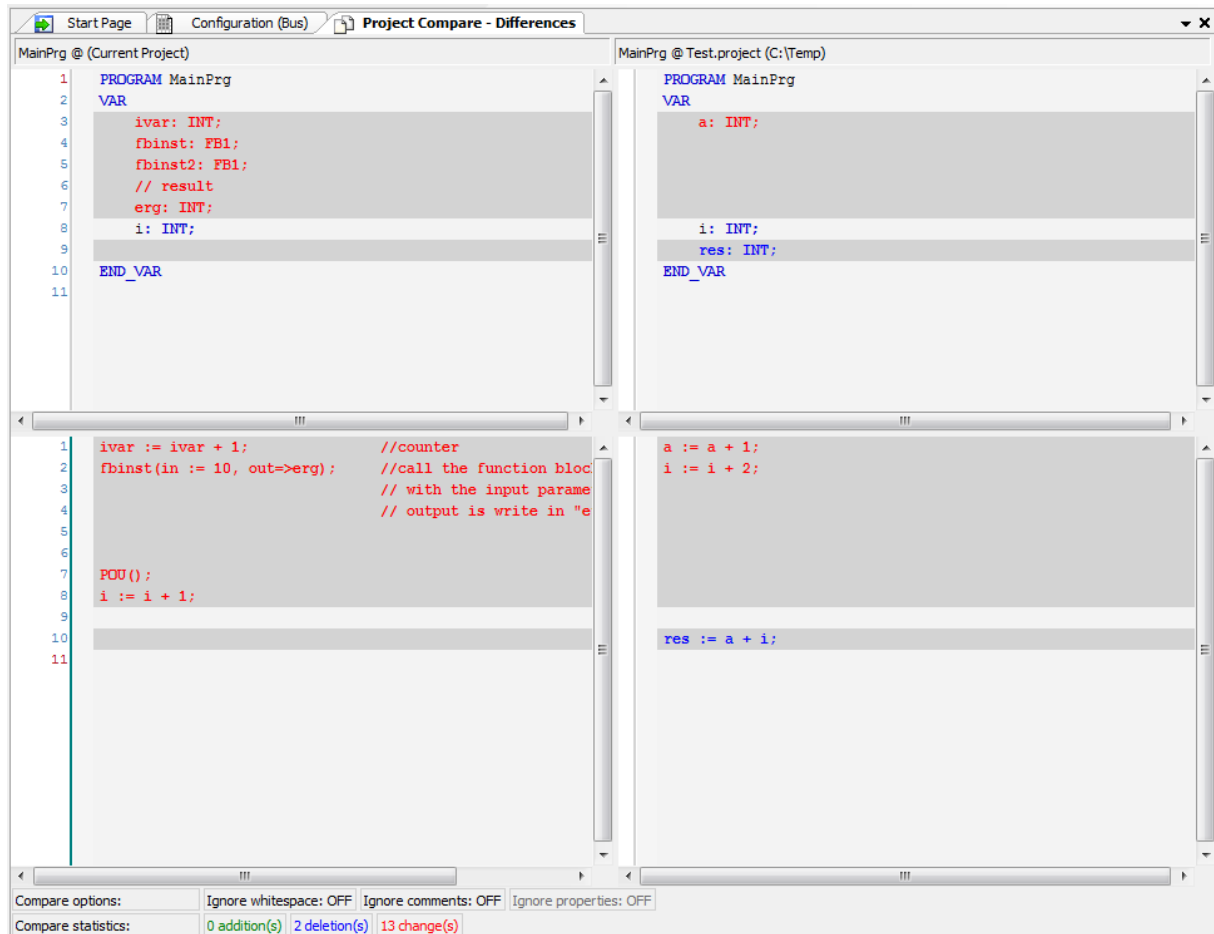


Figure 7-57. Example of Detailed Comparison Result

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:

- Logon
- Logoff
- Permissions

Logon

Symbol: 

This command opens the *Logon* dialog for logging on to a project or library via a 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.

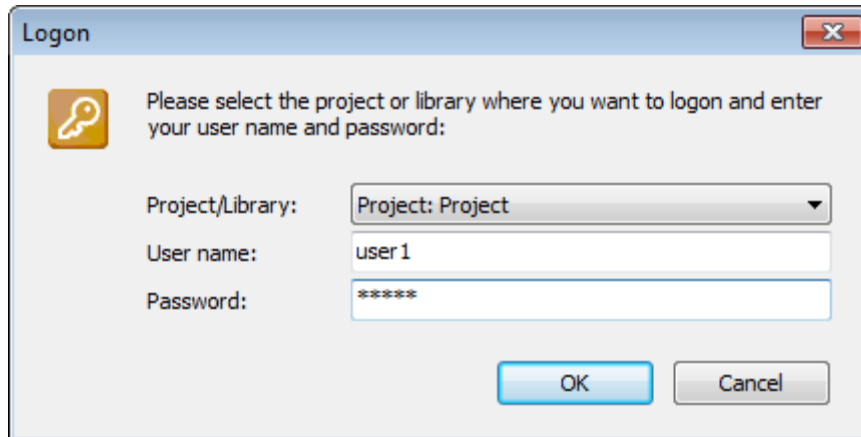


Figure 7-58. Logon

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 logon 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 *Logon* dialog will be opened, giving the possibility to log on with another user account provided with the appropriate rights:

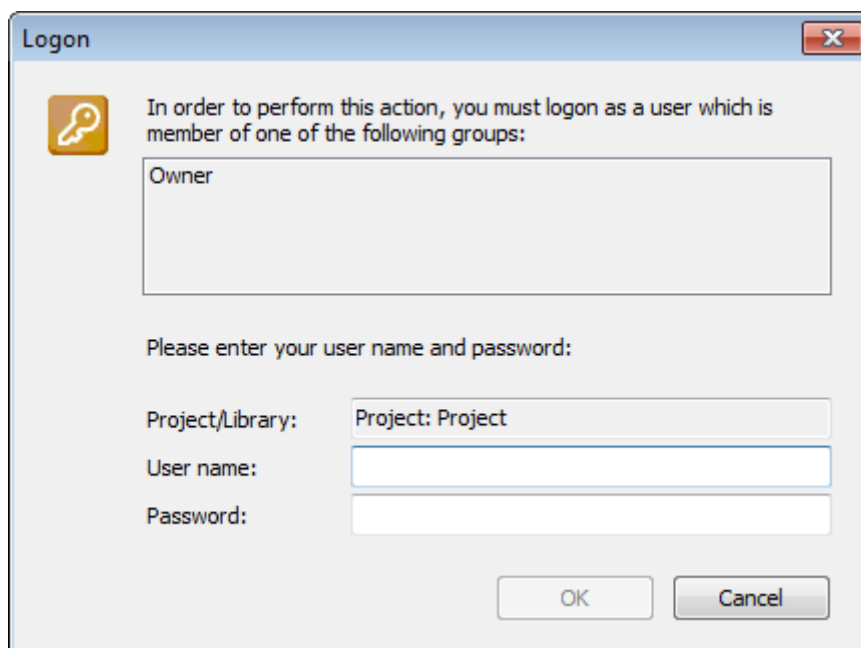


Figure 7-59. Logon Dialog on a Non-Permissible Action

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.

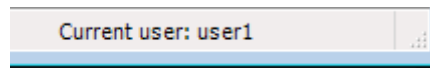


Figure 7-60. Status Line

Logoff

Symbol:

This command logs off the currently logged-on 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 logoff.

If the user currently is logged on to more than one project or referenced library (not necessarily with the same user account) a Logoff 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.

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.

Notice that any changes made in this dialog will be applied immediately.

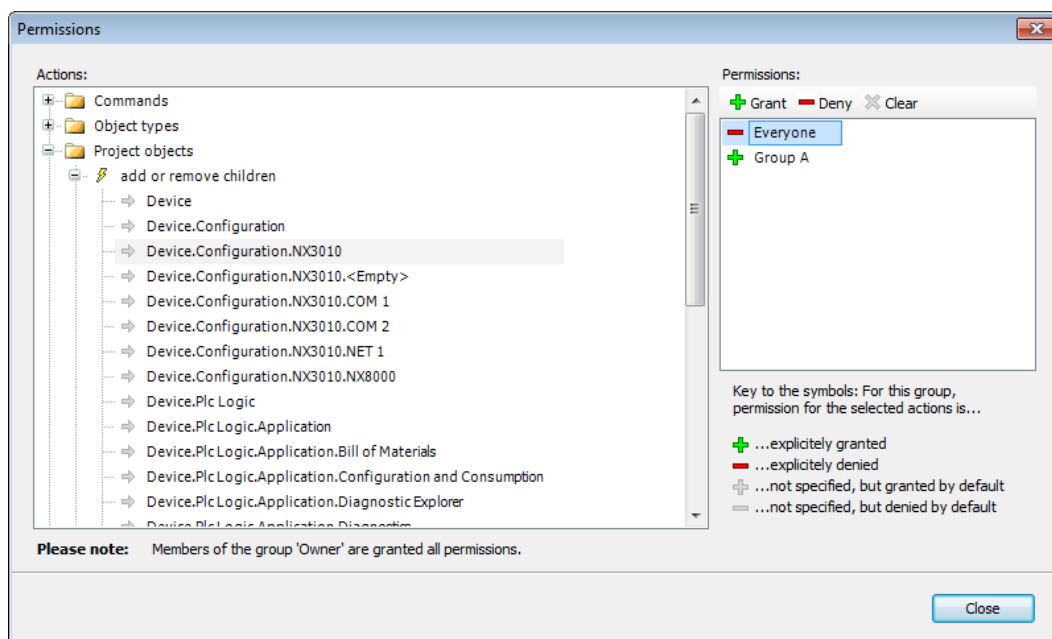


Figure 7-61. Permissions

The *Actions* window displays all possible rights that is all actions, which might be performed on any object of the current project. The tree is structured in the following way:

- : Top-level see the names of some categories, which have been set up just for the purpose of optical structuring the rights management. They are grouping concerning the execution of Commands, the configuration of User accounts and Groups, the creation of Object Types, the viewing, editing, removing and handling of child objects of Project Objects.
- : Below each category node there are nodes for the particular actions which might be performed on the command, user account, group, object type or project object. These nodes also only have optical function.

Possible actions:

 *Execute* (execution of a menu command).

 *Create* (creating a new object in the current project).

 *Add or Remove children* (adding or removing of "child" objects to an existing object).

 *Modify* (editing an object in an editor).

 *Remove* (deleting or cutting an object).

 *View* (viewing an object in an editor).

➔: Below each action node find the possible “targets”, that is project objects, of the respective action.

The *Permissions* window provides a list of all currently available user groups (except the *Owner* group) and a toolbar for configuring rights to a group.

Left to each group name one of the following icons indicates the currently assigned permission concerning the target, which is currently selected in the *Actions* window:

- : The action(s) currently selected in the *Actions* window are granted for the group.
- : The action(s) currently selected in the *Actions* window are denied for the group.
- : The right to perform the action(s), which are currently selected in the *Actions* window, has not been granted explicitly, but is granted by default, for example because the corresponding right has been granted to the “father” object. Basically, this is the default setting for all rights, which not explicitly have been configured.
- : The right to perform the action(s) currently selected in the *Actions* window has not been designed explicitly, but is denied by default, for example in case because the corresponding right has been assigned to the “father” object.

If currently multiple actions are selected in the *Actions* window, which do not have unique settings referring to the currently selected group, no icon will be displayed.

To configure the rights for a group select the desired action(s) in the *Actions* window and the desired group in the *Permissions* window. Then use the appropriate button in the toolbar of the *Permissions* window:

-  **Grant**: Explicit granting.
-  **Deny**: Explicit denying.
-  **Clear**: The currently granted right for the action(s) currently selected in the *Actions* window will be deleted, that is set back to the default).

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/IL Editor Commands
- CFC Editor Commands
- SFC Editor Commands

The ST language editor does not have a specific menu.

FBD/LD/IL Editor Commands

The “FBD/LD/IL” category provides commands for working in the FBD/LD/IL Editor, which is a collective editor for the languages FBD (Function Block Diagram), LD (Ladder Diagram) and IL (Instruction List).

Note: For inserting elements in the FBD and LD editor view regard also the possibility to drag them directly from the toolbox or from another position within the editor.

Insert Network

Symbol: 

Default Shortcut: <CTRL>+<I>

This command is used to insert a network in the FBD, LD or IL 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.

Notice that in contrast to the previous version the network element is also used in the IL editor.

NOTE: The display of the elements in a LD/FBD/IL network is defined in the **FBD, LD and IL Editor**.

Insert Network Below

Symbol: 

Default Shortcut: <CTRL>+<T>

This command is used to insert a network in the FBD, LD or IL editor.

If the cursor currently is placed within an existing network, the new network will be inserted immediately below 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.

Notice that in contrast to the previous version the network element is also used in the IL editor.

NOTE: The display of the elements in a LD/FBD/IL network is defined in the **FBD, LD and IL Editor**.

Toggle Network Comment State

Symbol: 

Default Shortcut: <CTRL>+<O>

This command can be used in FBD, LD or IL editor to comment out a network to set it back from comment to normal state. The command will effect 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.

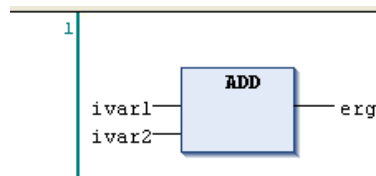


Figure 7-62. Normal State

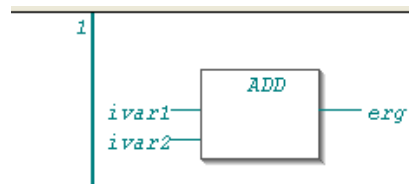


Figure 7-63. Comment State

NOTE: Concerning the view options for the components of a LD / FBD network notice the settings in the **FBD, LD and IL Editors**.

Insert Assignment

Symbol: VAR]

Default Shortcut: <CTRL>+<A>

This command is used to place an assignment in a LD, FBD or IL editor.

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 "???".

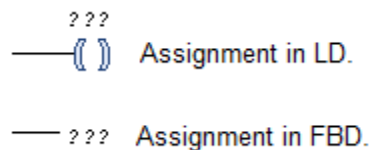


Figure 7-64. Assignment in LD/FBD

To define the assignment select the "???" and replace it by the name of the variable that is to be assigned. The Input Assistant can be used for this purpose.

In IL an assignment is programmed via the "LD" and "ST" operators.

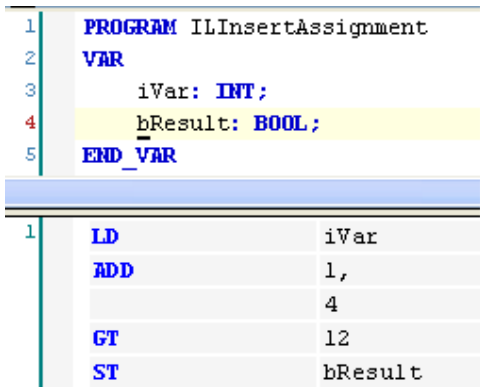


Figure 7-65. Assignment in IL

NOTE: Concerning the view options for the components of a LD / FBD network notice the settings in the **FBD, LD and IL Editors**.

Insert Box

Symbol:

Default Shortcut: <CTRL>+

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. In the IL editor the corresponding instructions will be inserted.

As soon as you choose the command, the *Input Assistant* dialog will open, providing the appropriate categories of POU's. 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.

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.

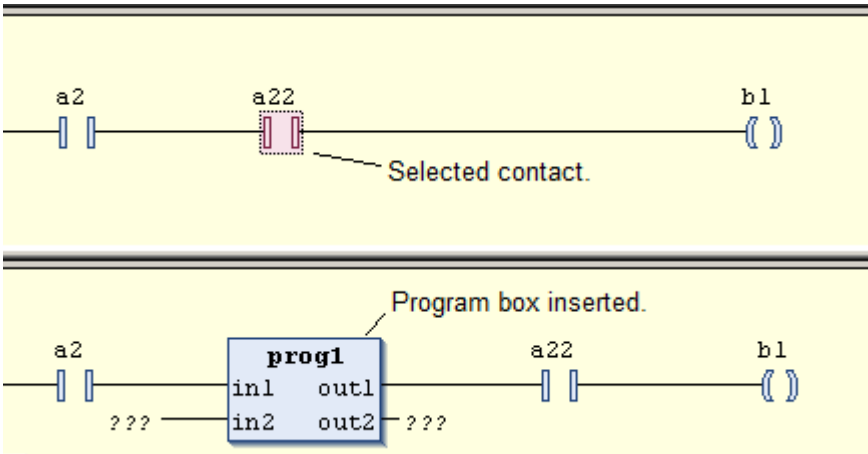


Figure 7-66. Insert Function Block

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 ⇄.

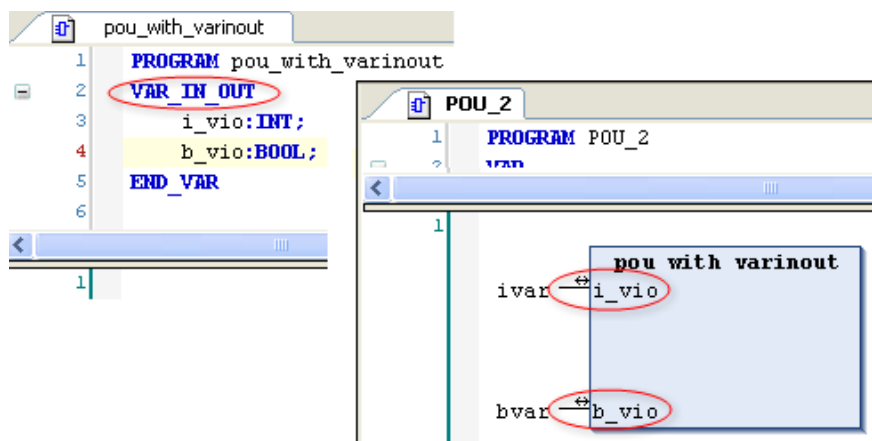


Figure 7-67 Display of VAR_IN_OUT Parameters

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**.

IL Editor Specific Characteristics

In IL also a "box" can be inserted at any desired line. If the *Input Assistant* is used with option *Insert with arguments*, the chosen POU will be displayed in form of a CAL instruction and the respective input and output parameters of the chosen box element.

1	CAL	TONinst{	
		IN: = ???,	
		PT: = ???,	
		ET=> ???)	

Figure 7-68. Box Inserted in IL

Box element "TON" has been inserted; instance “TONinst” has been defined locally. Input parameters: IN, PT. Output parameter: ET.

Insert Empty Box

Symbol: 

Default Shortcut: <CTRL>+<SHIFT> +

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.

In the IL editor the corresponding instructions will be inserted

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.

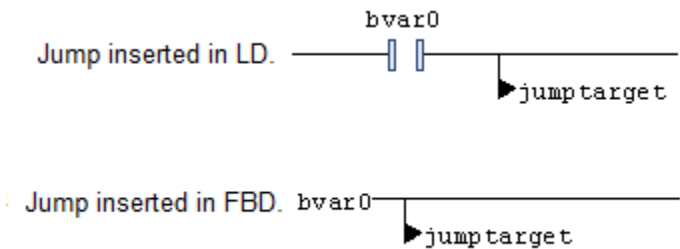


Figure 7-69. Jump Inserted

```
1 PROGRAM IL
2 VAR
3     bVar: BOOL;
4     bVar2: BOOL;
5 END_VAR

1 LD      bVar
  JUMPC   jumptarget
2
jumptarget:
1 LD      bVar2
```

Figure 7-70. Programmed Jump via JMP Operator (IL Editor)

If a JMP operator, that has been inserted in IL without preceding LD is converted to LD, the operator "???" will be inserted.

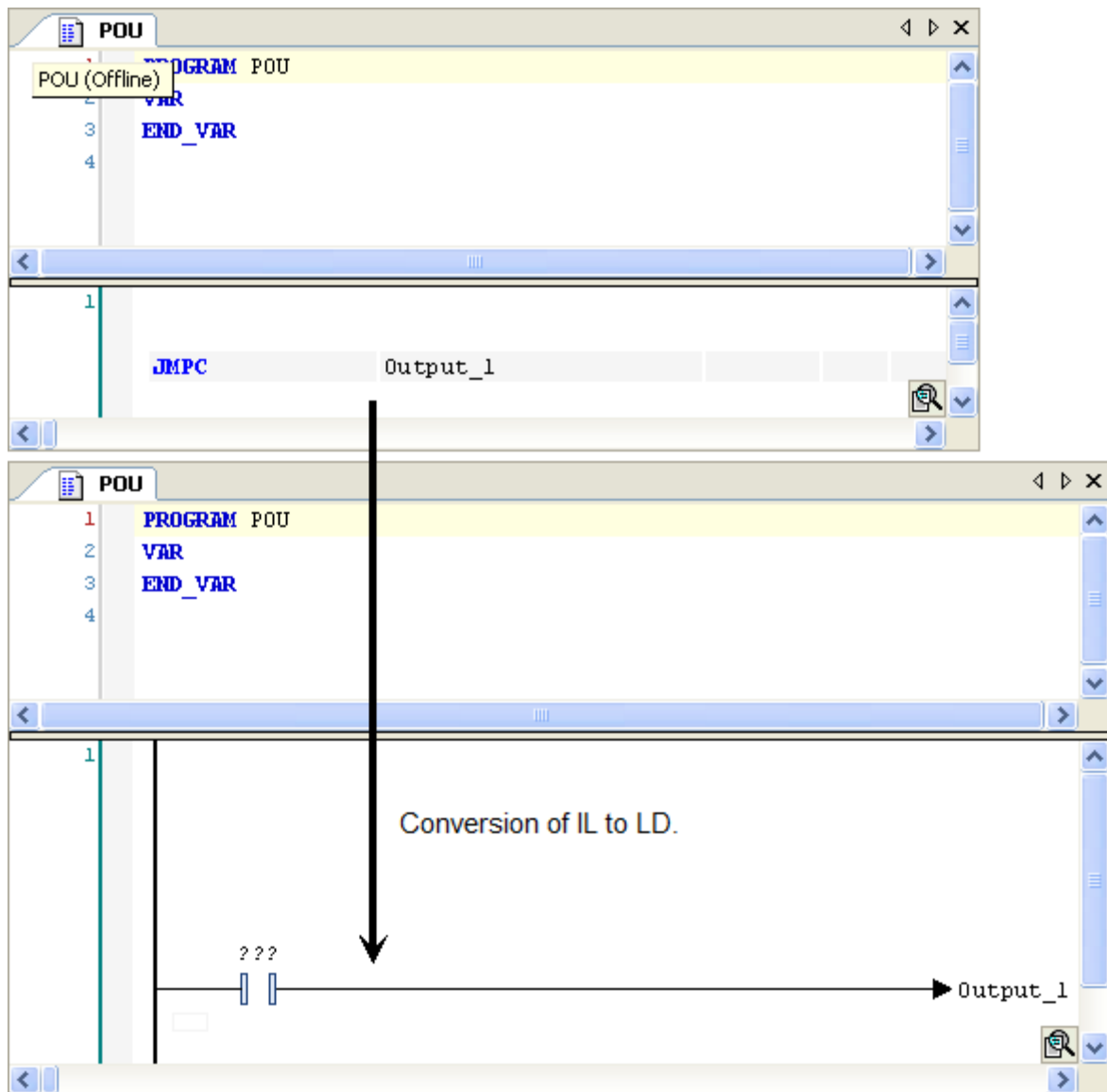


Figure 7-71. JMP Operator Conversion

NOTE: Concerning the view options for the components of a FBD/LD/IL network notice the settings in the **FBD, LD and IL Editor**.

Insert Label

Symbol: 

This command inserts a label field into the currently selected network. The default text string can be modified according to your demands:

- 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.

Insert Return

Symbol: 

This command inserts a RETURN instruction.

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/IL network notice the settings in the **FBD, LD and IL Editor**.

IL

In IL a Return instruction will be inserted.

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. It is not available in the IL 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/IL network notice the settings in the **FBD, LD and IL Editor**.

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**.

*Insert Set Coil***Symbol:** 

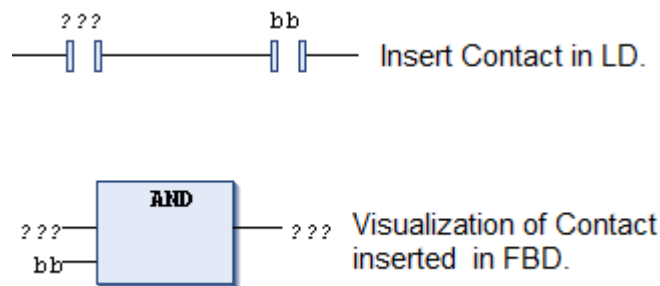
This command is used to insert a set coil. Alternatively, the combination of the commands *Insert Coil* and *Extras Set/Reset* can be used to insert a set coil.

*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.

*Insert Contact***Symbol:** **Default Shortcut:** <CTRL>+<K>

This command inserts a contact in a LD network. It is not available in the FBD and IL editor but will be converted appropriately when switching views.

**Figure 7-72. Insert Contact**

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>+KEY and then use the command.

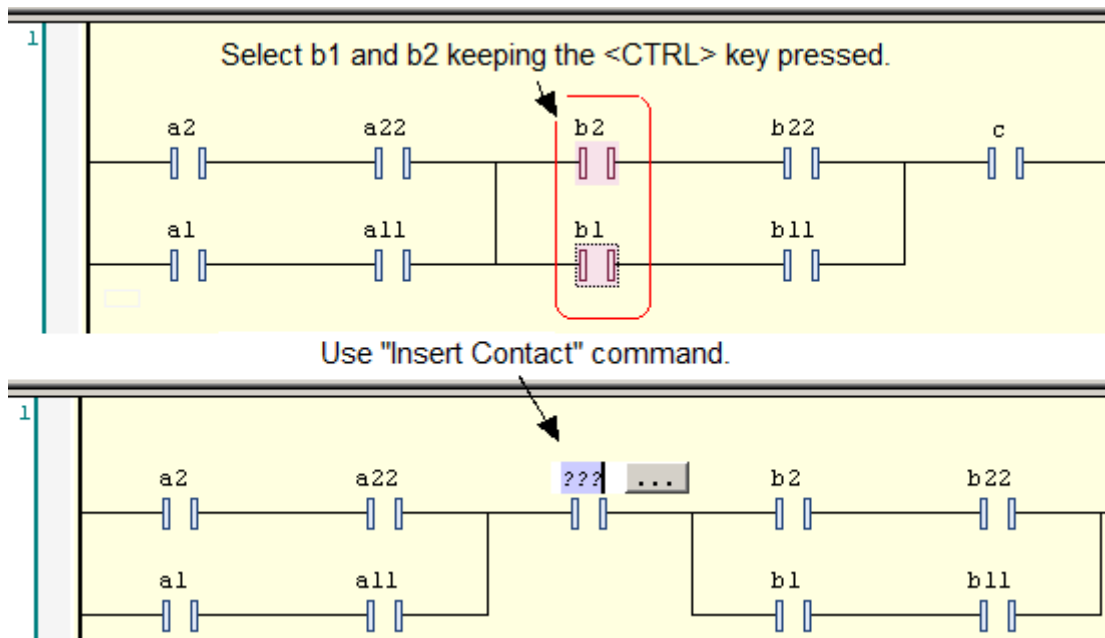


Figure 7-73. Inserting Contacts

Notice also the commands **Insert Contact (right)**, **Insert Contact Parallel (above)**, **Insert Contact Parallel (below)**.

A new contact element is preset with the text “???”. You can select this text by a mouse-click and replace it by the name or address (depending on the current settings in the FBD/LD and IL Options dialog) of the desired variable or the desired constant. The *Input Assistant* can be used for this purpose.

NOTE: Concerning the view options for the components of a FBD/LD/IL network notice the settings in the **FBD, LD and IL Editor**.

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.

Insert Contact (right)

Symbol:

Default Shortcut: <CTRL>+<D>

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.

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.

*Insert Negated Contact Parallel (below)***Symbol:** 

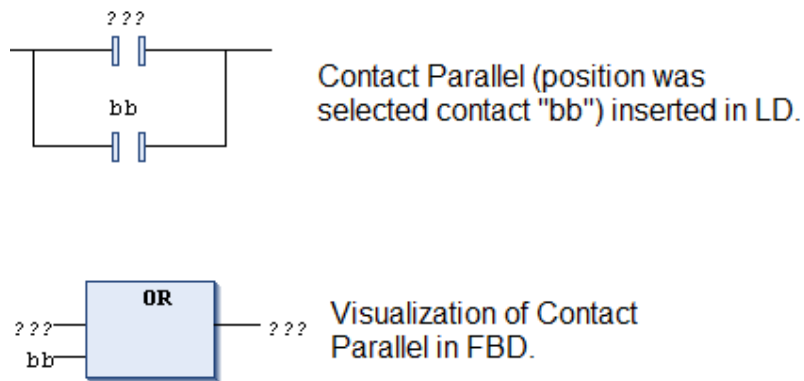
This command is used to insert a negated parallel contact. Alternatively, a combination of the commands *Insert Contact Parallel (Below)* and *Negation* can be used to insert a negated parallel contact.

*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 and IL 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 7-74. View of Inserted Parallel Contact***Paste Contacts: Paste below*

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.

*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.

*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.

Insert IL Line (below)

Symbol: .

This command is only available in the IL editor. It is used to insert a further instruction line below the currently selected one.

Dele IL Line

Symbol: .

Default Shortcut: <CTRL>+<DEL>

This command is only available in the Instruction List editor. It is used to delete the line where currently the cursor is positioned.

Negation

Symbol:  (FBD),  (LD)

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. It is not available in the IL editor, where the corresponding modifiers have to be used appropriately.

ATTENTION:

If view is switched from FBD or LD to IL view and back, the negations of some constructs might be set back because an un-ambiguous conversion is not possible.

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:

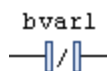


Figure 7-75. Negated Contact

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 or RETURN instruction 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/IL network notice the settings in the **FBD, LD and IL Editor**.

Edge Detection

Symbol:  (FBD),  (LD)

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.

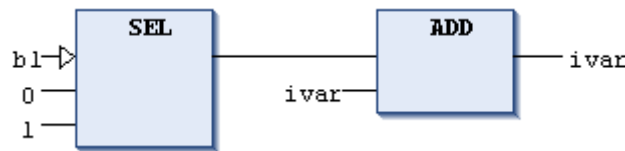


Figure 7-76. Edge Detection at SEL Operator

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.

The command is not available in IL editor. A network containing an edge detection will be kept unmodified after conversion from FBD/LD to IL.

NOTE: Concerning the view options for the components of a FBD/LD/IL network notice the settings in the **FBD, LD and IL Editor**.

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). It is not available in the IL editor, where the corresponding modifiers have to be used appropriately.

With multiple executions of the command, the output will alternate between set, reset, and normal output.

NOTE: Concerning the view options for the components of a FBD/LD/IL network notice the settings in the **FBD, LD and IL Editor**.

Set Output Connection

Symbol: 

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.

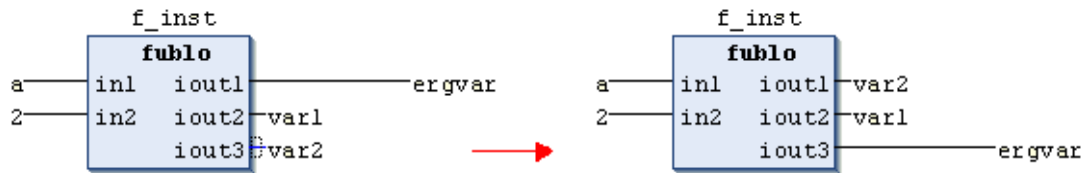


Figure 7-77. Change Output Connection

NOTE: Concerning the view options for the components of a FBD/LD/IL network notice the settings in the **FBD, LD and IL Editor**.

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.

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.

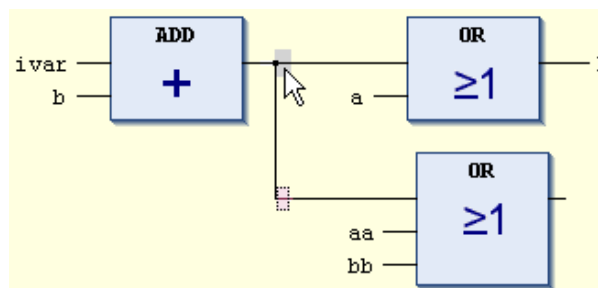


Figure 7-78. Sub-network Markers in FBD Network

Each subnetwork can get another branching, thus multiple branches and thus a widely ramified construction of “sub-networks” is possible within the main network.

For adding branches to branches see the commands: **Insert Branch above** and **Insert Branch below**.

The command is not available in IL editor. Networks with branch elements cannot be converted to IL.

NOTE: Concerning the view options for the components of a FBD/LD/IL network notice the settings in the **FBD, LD and IL Editor**.

Insert Branch Above

Symbol:

This command is available for inserting another branch within an existing one above the current cursor position.

Insert Branch Below

Symbol:

This command is available for inserting another branch within an existing one below the current cursor position.

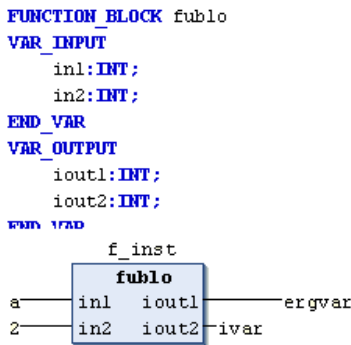
Update Parameters

Default Shortcut: <CTRL>+<U>

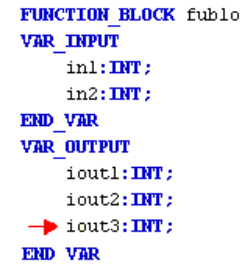
This command can be used in FBD, LD or IL editor to update the parameters (inputs, outputs) of a box, which is already inserted in a network, after having changed its interface for example by adding an output.

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.

1. Original version of the function block with two outputs.



2. Function block modified, added output.



3. Updated function block by "Update Parameters" in the FBD network.

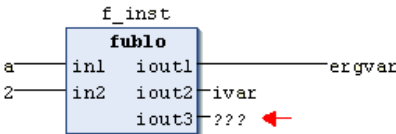


Figure 7-79. Update Parameters in FBs

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.

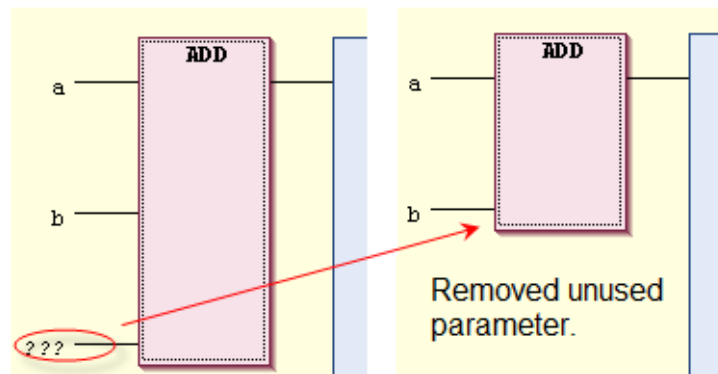


Figure 7-80. Remove Unused Parameters in FBs

View as function block diagram (FBD)

Default Shortcut: <CTRL>+<1>

This command is available if you currently are working in the Ladder Diagram (LD) or Instruction List (IL) view of the POU. It can be used in offline and online mode.

The LD networks respectively the instruction list 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 unambiguously.

To switch back to the LD view use command *View as ladder logic*.

To switch back to the IL view use command *View as instruction list*.

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/IL network notice the settings in the **FBD, LD and IL Editor**.

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 unambiguously.

FBD elements, which cannot be displayed as LD elements (for example XOR), will be displayed as FBD boxes within a LD network.

To switch back to the LD view use command *View as ladder logic*.

To switch back to the IL view use command *View as instruction list*.

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/IL network notice the settings in the **FBD, LD and IL Editor**.

View as instruction list (IL)

Default Shortcut: <CTRL>+<3>

This command is available if you currently are working in the Function Block Diagram (FBD) or Ladder (LD) view of the POU. It can be used in offline and online mode.

The FBD and LD networks will be converted to an instruction list.

NOTE: There are some elements, which cannot get converted and thus the respective network will remain in the original editor view. In case of syntax errors also no conversion is possible; a corresponding error message will be generated. Some constructs might not be convertible unambiguously and thus will be "normalized" after a conversion from IL to FBD/LD and back. This concerns negations and explicit/implicit assignment of function block in- and outputs

To switch back to the FBD view use command *View as function block diagram*.

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/IL network notice the settings in the **FBD, LD and IL Editor**.

CFC Commands

The commands of this category are available for programming in the CFC editor.

Available commands:

- Edit Working Sheet
- Negate
- EN/ENO
- Set/Reset (None)
- Set/Reset (S – Set)
- Set/Reset (R – Reset)
- Execution Order (Send to front)
- Execution Order (Send to back)
- Execution Order (Move up)
- Execution Order (Move down)
- Set Execution Order...
- Order by Data Flow
- Order by Topology

- Edit Parameters...
- Connect Selected Pins
- Reset Pins
- Remove Unused Pins

Edit Working Sheet

This command opens the *Edit Working Sheet* dialog for modifying the size of the working area of the current CFC.

The size of the working 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.

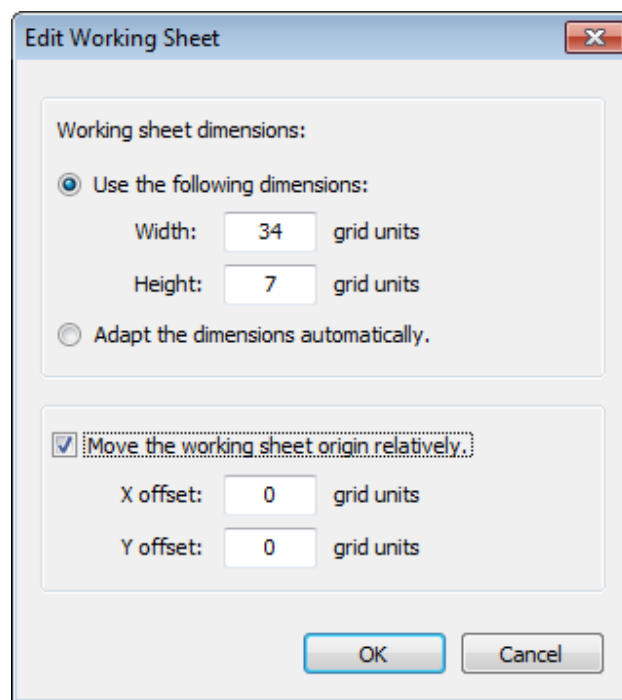


Figure 7-81. Edit Working Sheet

- *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.

Negate

Symbol: 

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.

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).

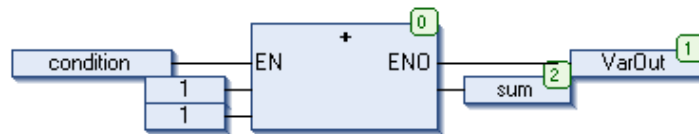


Figure 7-82. ADD-box with EN/ENO

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 7-83 shows how the ENO value can be used for further blocks.

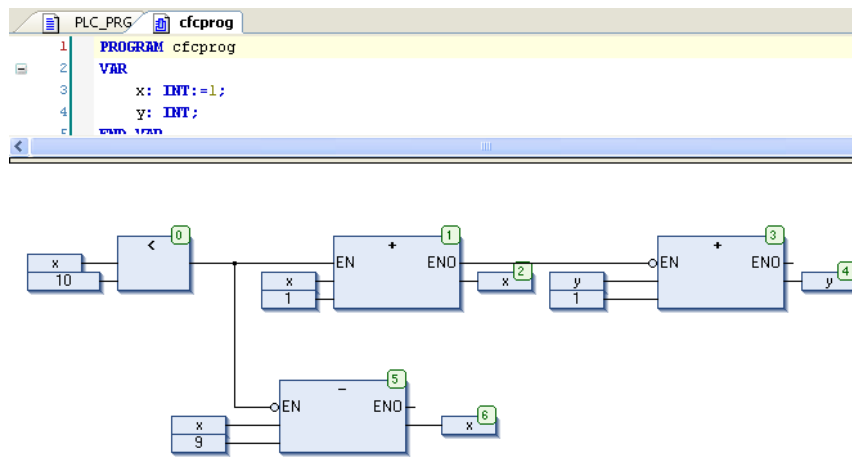


Figure 7-83. Use of EN/ENO

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 through the value range 1 to 10.

Set/Reset

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).

Reset

Symbol:

This command, by default part of the submenu *Set/Reset* of the CFC menu, 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 7-84. 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).

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 7-85. 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).

Execution Order

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.

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.

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.

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.

Order by Data Flow

This command is part of submenu *Execution Order* in the CFC menu. 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.

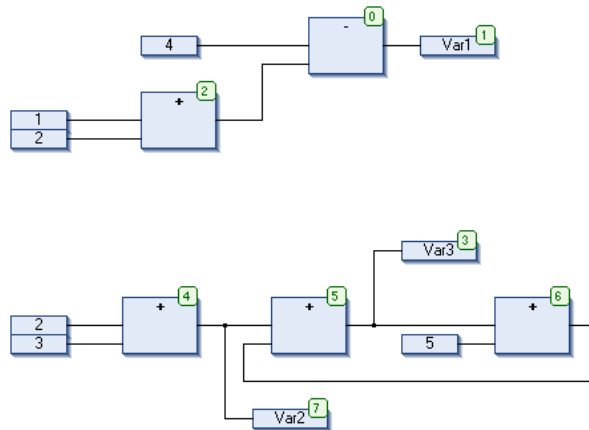


Figure 7-86. Order by Topology

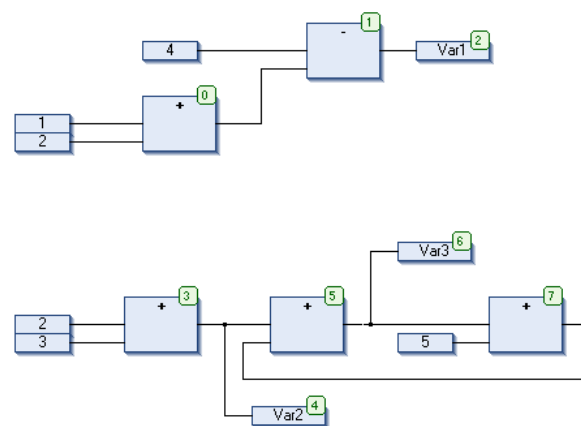


Figure 7-87. Processing According to Data Flow

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.

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:

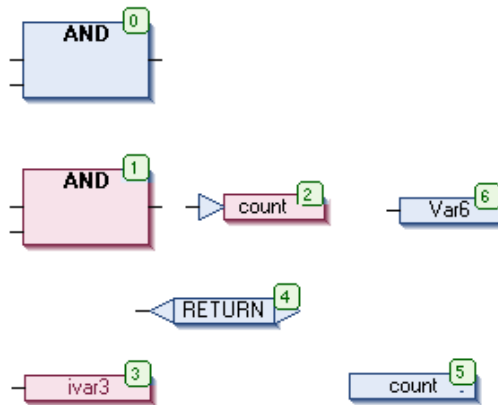


Figure 7-88. Sequence Before

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:

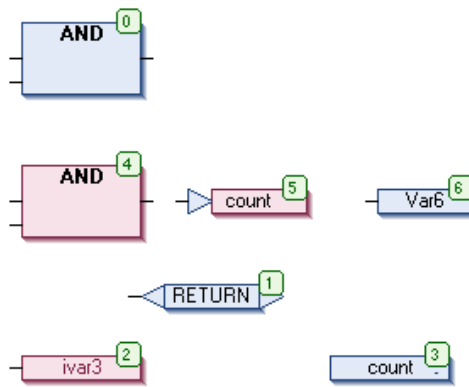


Figure 7-89. Sequence Afterwards

Also, a new element always will be inserted in the sequential processing list ahead of its topological successor.

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.

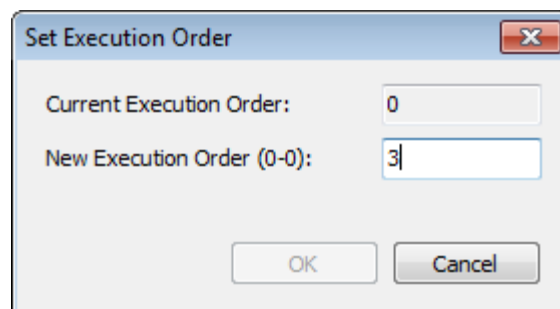


Figure 7-90. Set Execution Order

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.

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.

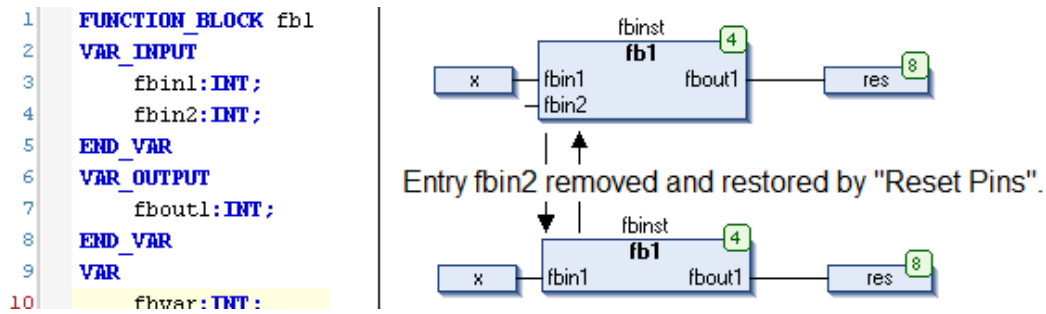


Figure 7-91. Reset Pins

Remove Unused Pins

Symbol:

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.

SFC Commands

The commands category SFC are available for programming in the SFC editor:

- Init step
- 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

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.

*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-... will be kept. See Figure 7-92 for examples.

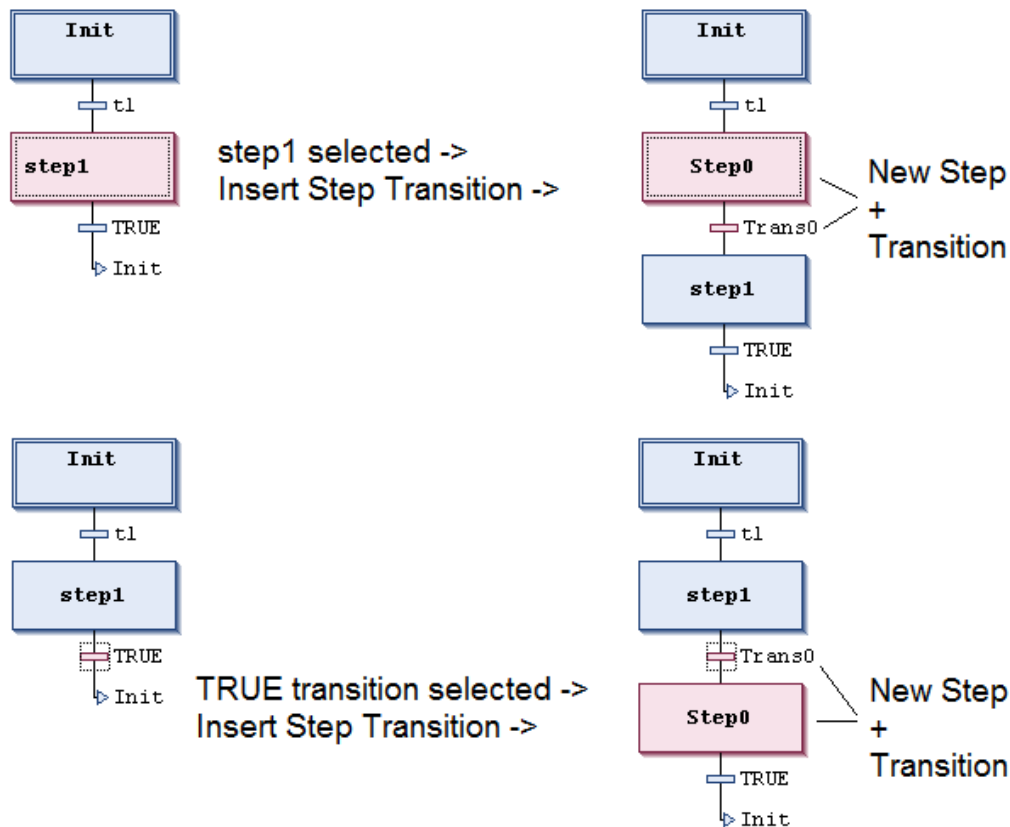


Figure 7-92. Step and Transition Inserted

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 to get it editable.

*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-... will be kept.

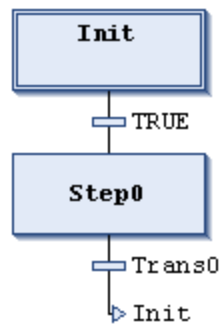


Figure 7-93. Step and Transition Inserted

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.

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.

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.

Insert Branch

Symbol:

This command is used in the SFC-Editor to insert a branch left to the currently selected element(s).

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*.

Example of Parallel Branch

In see a new parallel branch, created by command *Insert branch right* when *step11* was selected. Automatically a step (*Step2* in the example) gets inserted.

Processing in online mode: When “*t2*” is TRUE, *Step2* will be executed immediately after *step11* before “*t3*” is noticed. So both branches will be executed, in contrast to alternative branches.

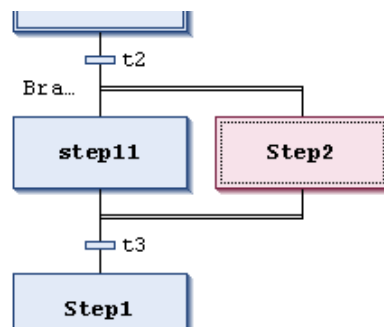


Figure 7-94. Parallel Branch

Example of Alternative Branch

In Figure 7-95 see a new alternative branch, created by command *Insert branch right* when transition “*t4*” was selected. Automatically a step (*Step32*) and a preceding and a subsequent transition (*t41*, *t42*) get inserted.

Processing in online mode: When *Step3* is active, the following transitions (*t4*, *t41*) 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.

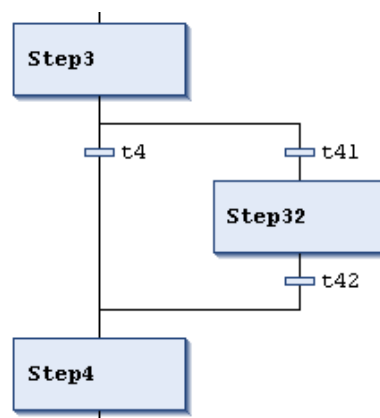


Figure 7-95. Alternative Branch

*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.

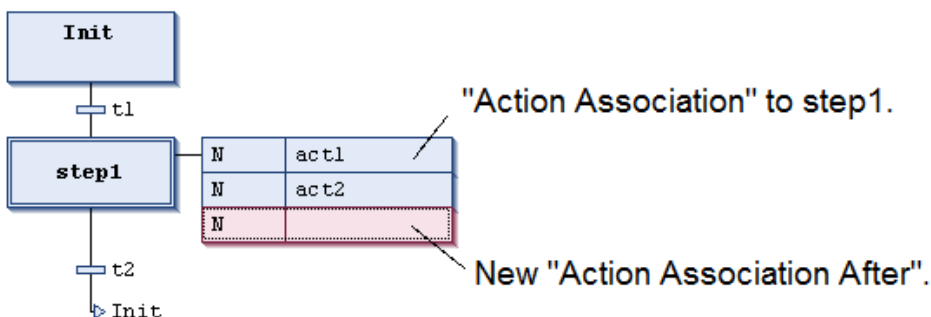
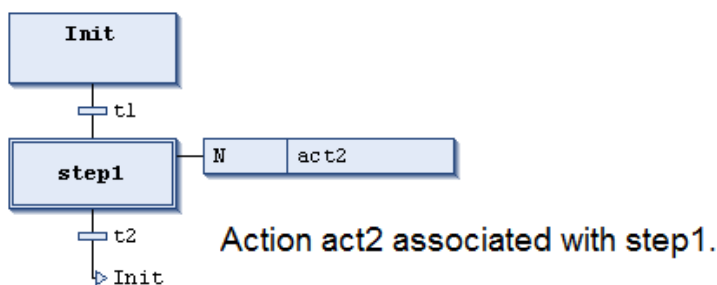


Figure 7-96. Actions Associated to a Step

*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.

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.

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.

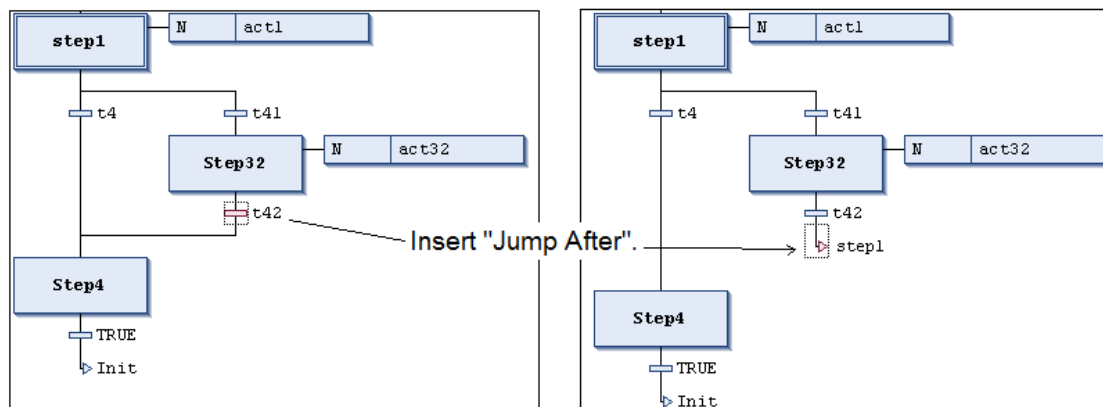


Figure 7-97. Insert Jump After

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*.

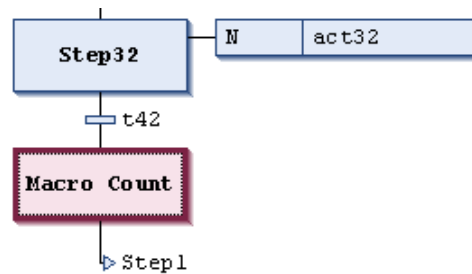


Figure 7-98. Macro Selected in SFC Diagram

*Insert Macro After***Symbol:**

This command is used in the SFC editor to insert a macro after the currently selected position in the diagram.

*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*.

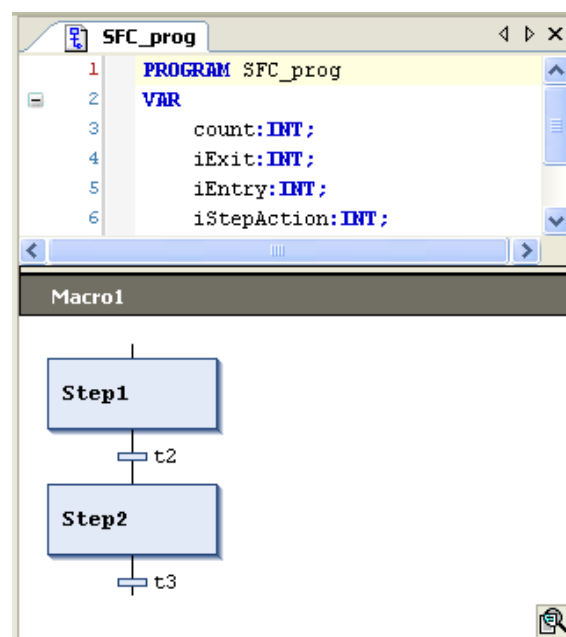


Figure 7-99. Macro Editor

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.

Build Menu

The command category *Build* provides commands for handling the project file.

These commands on the one hand serve to handle syntactical checks, either just on the changed objects or on all objects of the active application. On the other hand, the user can perform an offline code generation run in order to check for detected compile errors before downloading the code to the device.

The results each get displayed in the message window.

The *Clean* command serve to delete the compile information, which was created at the last, download - including generation of compilation code - on the target system. This for example will concern online change.

Available commands:

- Generate code
- Clean
- Clean all

Generate Code

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.

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.

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.

Online Menu

The *Online* menu provides commands apply in online mode.

Provides commands for controlling the application program on a real or on the simulation target system (device, PLC) after having logged in.

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.
- Online Change modifies the running application program and does not effect a restart process. Make sure that the new application code nevertheless will effect 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.

When calling the commands, which are marked by an asterisk (*) in the list below, the user will be prompted to confirm. Such a confirmation prompt will additionally appear for the commands, which are marked by two asterisks (**) in the list below, if option *Secure online mode* is activated in the communication settings of the respective device.

Available commands:

- Login
- Logout
- Create boot application
- Logoff current online user
- Download *
- Online Change *
- Source download to connected device
- Cluster Basic Configuration
- Reset warm *
- Reset cold *
- Reset origin*
- Simulation

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.

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.

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. For this purpose, a dialog box with the following text will open: “Application <application name> does not exist on device. Do you want to create it and proceed with download?”. The *Details* button in this dialog provides information on already applications on the PLC (corresponding to the information available in the Applications dialog of the Device Editor).
- 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. You will be asked whether that should be replaced via a dialog box with the following text: “Unknown version of Application “<Application>” on target: Do you want to perform a download and replace the application? Please carefully check all of the download procedures described on Nexto Series CPUs Manual.”. The *Details* button in this dialog provides some information (Project name, Last modification, IDE version, Author, Description) on the application in the IDE (integrated development environment = programming system) confronted to that on the controller.
- A version of the application is already available on the controller in RUN mode. You will be asked whether you really want to log in regardless and overwrite the currently running application. For this purpose a dialog box with the following text will open: “Warning: An unknown version of the application “<application name>” is currently in RUN mode on the PLC. However, do you want to download the latest code and replace the existing application? Please carefully check all of the download procedures described on Nexto Series CPUs Manual.”. The *Details* button in this dialog provides information on the application version in the programming system confronted to that on the controller.
- The application is already available on the controller but has been modified since the last download. You will be asked whether an online change should be done, or a login with loading the complete application code, or a login without any change of the running application. For this purpose, a dialog box with the following text will open: “Please carefully check all of the download procedures described on Nexto Series CPUs Manual. After this, choose an option to login: Login with online change, Login with download, Login without any change.” The *Details* button in the dialog provides information on the modified application in the programming system confronted to the previous version on the controller.

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.

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*.

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.

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.

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.

Create boot application

This command is available in online or offline mode for creating a boot project, also named “boot application”. The boot application serves the purpose to provide an application on the PLC, which will be loaded automatically when the PLC gets started.

Use in Online Mode

The boot application automatically will be stored as <application name>,.app on the PLC.

Use in Offline Mode

The standard dialog for saving a file will be opened. Select a folder, where you want to store the current application as boot application file. Automatically the file filter is “Boot application files” and extension “.app” will be applied to the defined file name. Therefore, you might store the boot project in any folder in order to transfer it to a PLC later. 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.

Logoff current online user

Performs the user logoff in the current project. For further information see **Logoff**.

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 <projectname>.<devicename>.<application>.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.

Online Change

ATTENTION:

Online Change modifies the running application program and does not effect a restart process. Make sure that the new application code nevertheless will effect 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.

This command is used to perform an online change on the currently active application.

Online change means that only the modified parts of an already running project will be re-loaded to the PLC. This is not possible after a *Clean all* or *Clean* application and code generation operation! The clean process removes the compile information, which is automatically stored at each, which is the basis on an online change.

During download the message view in category *Build* lists - among the usual information on the download process - also the changed interfaces, the concerned variables and all objects for which new code has been generated. If data locations change, a message box will hint at possible problems referring to the use of pointers.

Notice that an online change automatically will be offered when you are going to log in on a PLC with an application program, which is already running there, but has been modified since the last download.

Source download to connected device

This command is used to create and transfer a file from the current project for any device.

The command opens the *Select Device* dialog where the user must choose the network path to the PLC, as in the *Communication Settings* dialog. Select the corresponding item in the devices tree and press OK. This will set up a connection with the device and the source code will be sent in the form of a file.

The source code can be reloaded for the programmer in offline mode through the command *Source Upload*, on the *File* menu.

The default settings for the target device, content and time for sending the source code are set in the *Project Settings*, category *Source Download*.

Configuration of Basic Cluster

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.

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.

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.

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.

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 (☒ *Simulation*), 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 Figure 7-100 for an example: *Login* has just been performed for the currently active application.

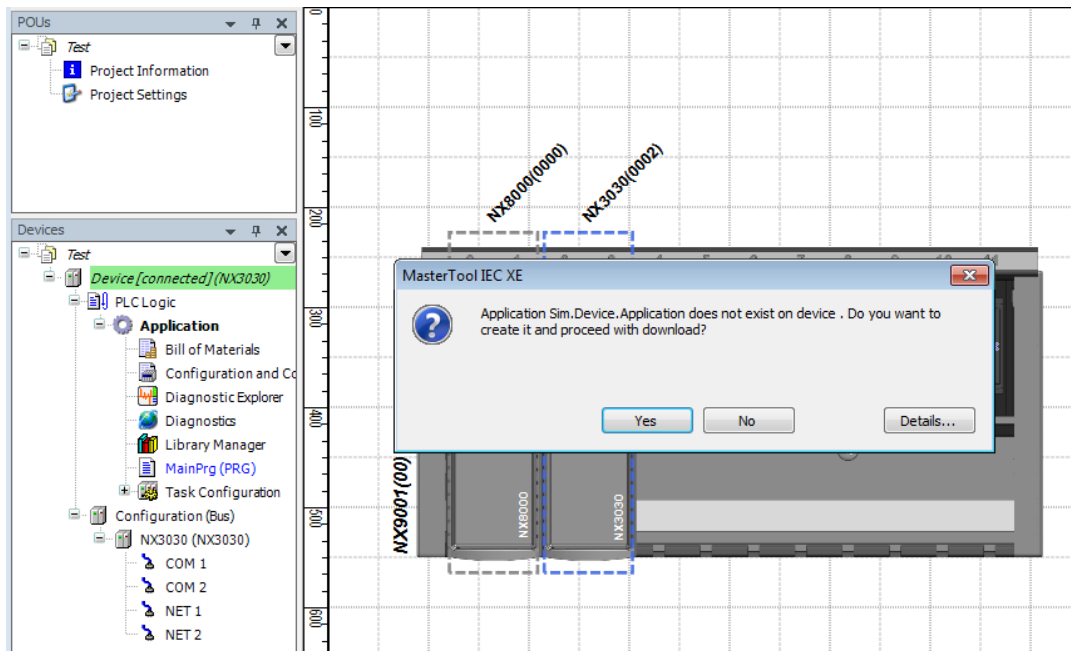


Figure 7-100. Login to the Simulator Target

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**.

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...
- Toggle Breakpoint
- Step Over
- Step Into
- Step Out
- Run to Cursor
- Set next statement

- Show next statement
- Write values **
- Force values **
- Unforce values
- Add All Forces To Watchlist
- Display Mode

Start

Symbol: 

Default Shortcut: <F5>

This command starts the application program on the device (PLC).

Stop

Symbol: 

Default Shortcut: <SHIFT> +< F8>

This command stops the application program on the device (PLC).

Breakpoints

The category “Breakpoints” provides commands for handling breakpoints for debugging purposes. These commands allow adding, removing or changing the breakpoints.

Symbols indicating the breakpoints status:

Breakpoint enabled ()

Breakpoint disabled ()

Stop at breakpoint in online mode ()

Breakpoints can be defined to work as halt positions when processing an application program for example for debugging purposes.

Further on there is the possibility to execute the program in defined steps (stepping).

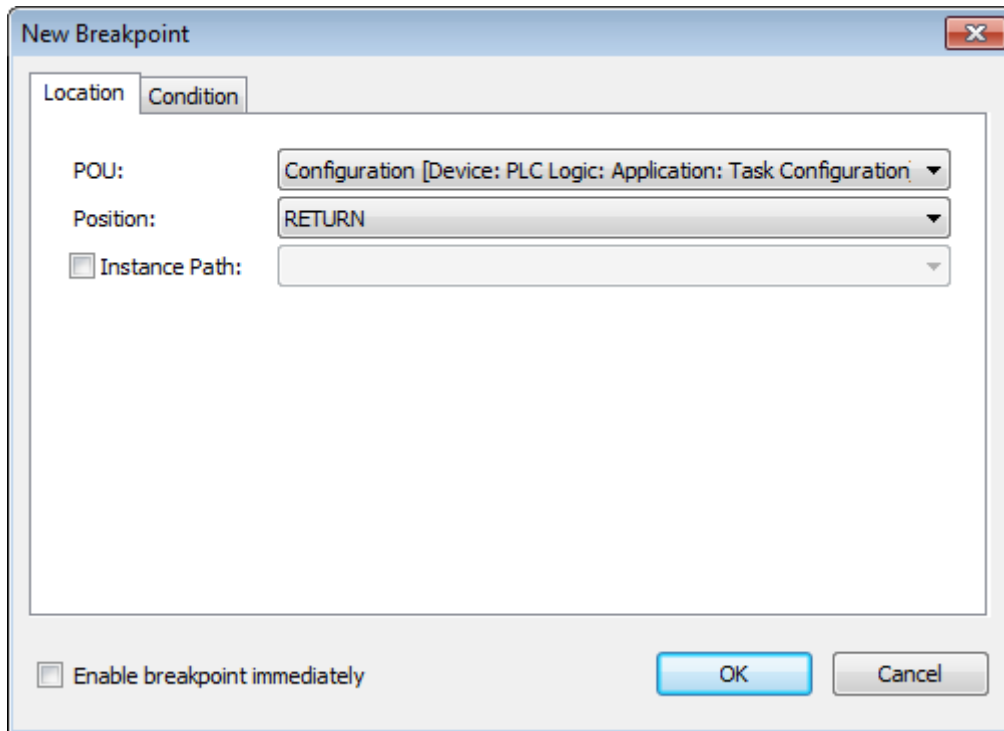
NOTE: When an application is stopped on a Breakpoint, the module diagnostic variables will not be updated until a MainTask cycle is completed. For this, it is necessary to run the application through all Breakpoints until it stops again on the first breakpoint present. When it stops on this Breakpoint the diagnostics will be updated

New Breakpoint

Symbol: 

This command is used to insert a new breakpoint in one of the POU's. 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**.

Location**Figure 7-101. New Breakpoint Dialog, Location**

- *POU*: The selection list offers all POU's 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.
- *Instance Path*: If the currently selected POU is a function block and this option is deactivated, the breakpoint will be set in the implementation body of the POU. If you want to set the breakpoint in an instance, activate the option and select the desired instance.

Condition

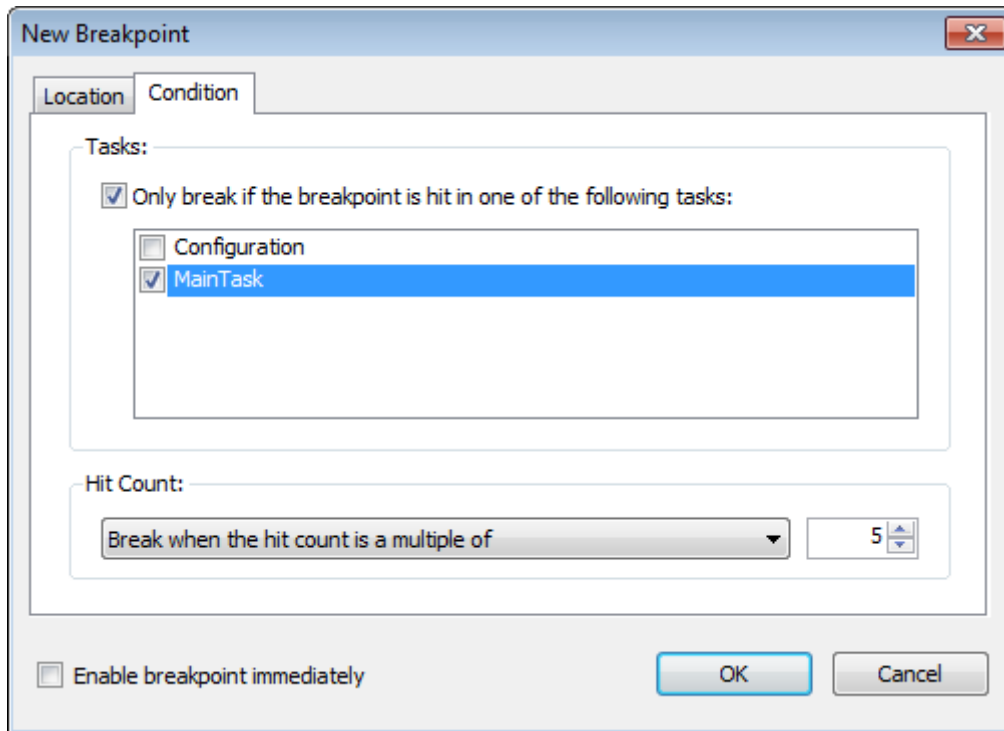


Figure 7-102. New Breakpoint Dialog, Condition

- *Only break if 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...*

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.

Breakpoint Symbols

Figure 7-103, Figure 7-104 and Figure 7-105 show the possible symbols assumed by breakpoints.

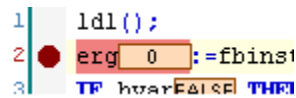


Figure 7-103. Breakpoint in Online Mode

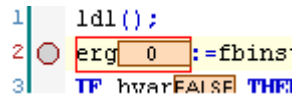


Figure 7-104. Disabled Breakpoint

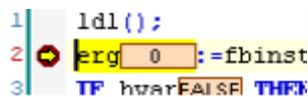


Figure 7-105. Program Stop at Breakpoint

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.

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.

Step Into

Symbol:

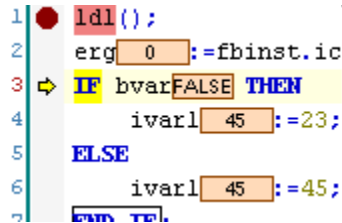
Default Shortcut: <F8>

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*.



```

1 | ● ldl();
2 |   erg 0 :=fbinst.ic
3 | → IF bvarFALSE THEN
4 |     ivarl 45 :=23;
5 |   ELSE
6 |     ivarl 45 :=45;
7 |   END IF;

```

Figure 7-106. Step Into

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.

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.

NOTE: One way to perform one determined task at once is adding a breakpoint inside the beginning of the POU associated with the task. Then you can use the *Run* or *Run To Cursor* to get a complete cycle execution.

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.

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.

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 (watchlist);
- 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.

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 7-107 presents the sequence of processing in a cycle of application program.

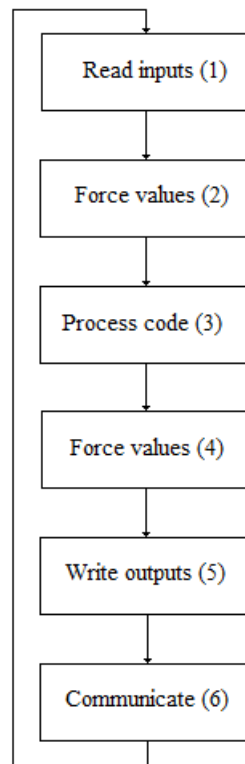


Figure 7-107. Sequence of processing in a cycle

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 actives 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:

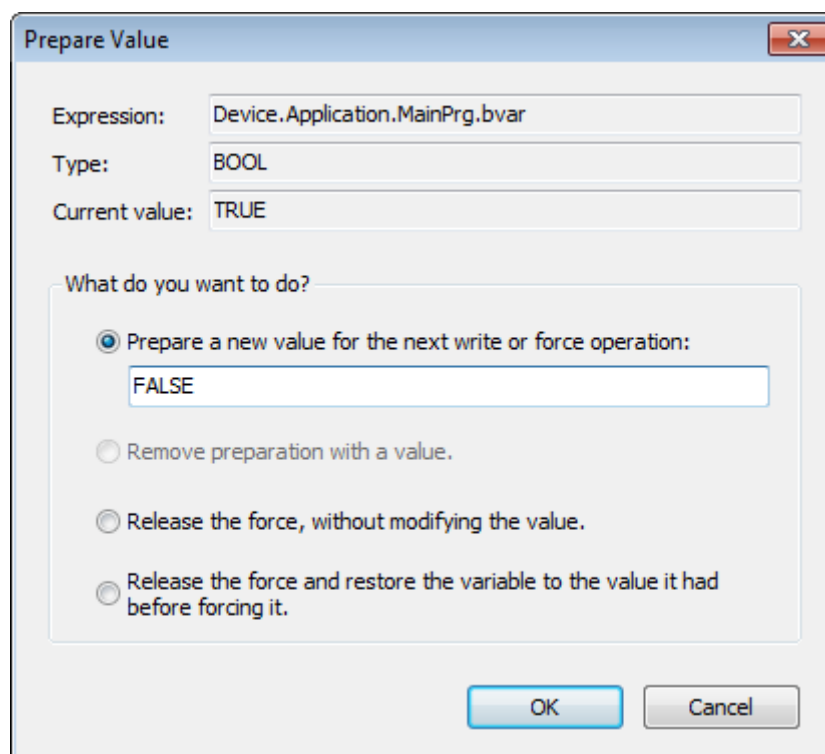
- In a watch view defined in the project, containing a list of variables to be monitored (watch list).

- 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 /LD/IL editor.
- A forced value is indicated with a symbol **F**.

Device.Application.MainPrg				
Expression	Type	Value	Prepared value	Comment
ivar	INT	0		
fbinst	FB1			
fbinst2	FB1			
erg	INT	F 100		result
bvar	BOOL	F TRUE		
i	INT	0		

Figure 7-108. Forced Values in the Declaration Editor of a POU (Online View)

Prepare Value Dialog



The 'Prepare Value' dialog box is shown with the following fields and options:

- Expression:** Device.Application.MainPrg.bvar
- Type:** BOOL
- Current value:** TRUE
- What do you want to do?**
 - ☒ Prepare a new value for the next write or force operation: FALSE
 - ☐ Remove preparation with a value.
 - ☐ Release the force, without modifying the value.
 - ☐ Release the force and restore the variable to the value it had before forcing it.
- Buttons:** OK, Cancel

Figure 7-109. 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/IL editor.

Device.Application.MainPrg				
Expression	Type	Value	Prepared value	Comment
ivar	INT	0		
fbinst	FB1			
fbinst2	FB1			
erg	INT	F 100	25	result
bvar	BOOL	F TRUE	FALSE	
i	INT	0		

Figure 7-110. Prepared Value Field

The following information on the currently concerned variable is displayed:

- *Expression*: path of the variable (“Device.Application.MainPrg.ivar”, for example).
- *Type*: data type (“DWORD” 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.

There is a maximum limit on the number of allowed forces on MasterTool IEC XE. 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.

Unforce Values

Default Shortcut: <ALT>+<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 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.

Add All Forces to the Watchlist

This command is available in online mode when one of the watch views of watch 1, 2, 3 or 4 is active. It serves to add all currently prepared or already forced variables of the active application to this watchlist. 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.

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*
- *Hexadecimal*

Tools Menu

This menu provides commands for installation/uninstallation of libraries and devices, customization and project options, environment, editors, etc.

Available commands:

- Library Repository
- Install Library
- Device Repository
- Install Device
- Options

Library Repository

Symbol: 

The Library Repository is accessed through the *Tools* menu and opens the *Library Repository* dialog.

NOTE: The *Library Repository* dialog is only available if predefined feature sets chosen by the user are *Professional* or the option *Enable repository dialog* is enabled. For further information about features, see **Features**.

NOTE: The *Library Repository* dialog, which can also be accessed through the editor of *Library Manager* will be available depending on the type of the installation. A repository of libraries is the database for libraries that have been installed on the local system, aiming at a later inclusion in projects of MasterTool IEC XE. A library project *. library, stored in a repository cannot be opened out of this for editing or viewing in programmer.

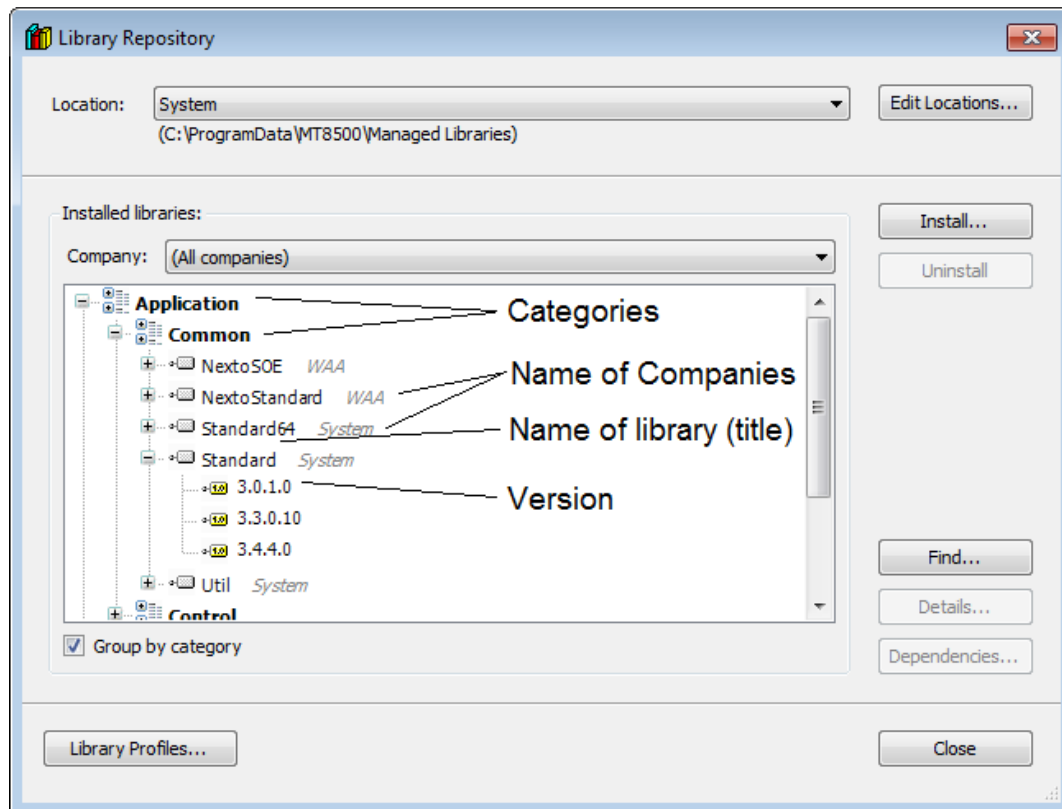


Figure 7-111. Library Repository

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)

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.

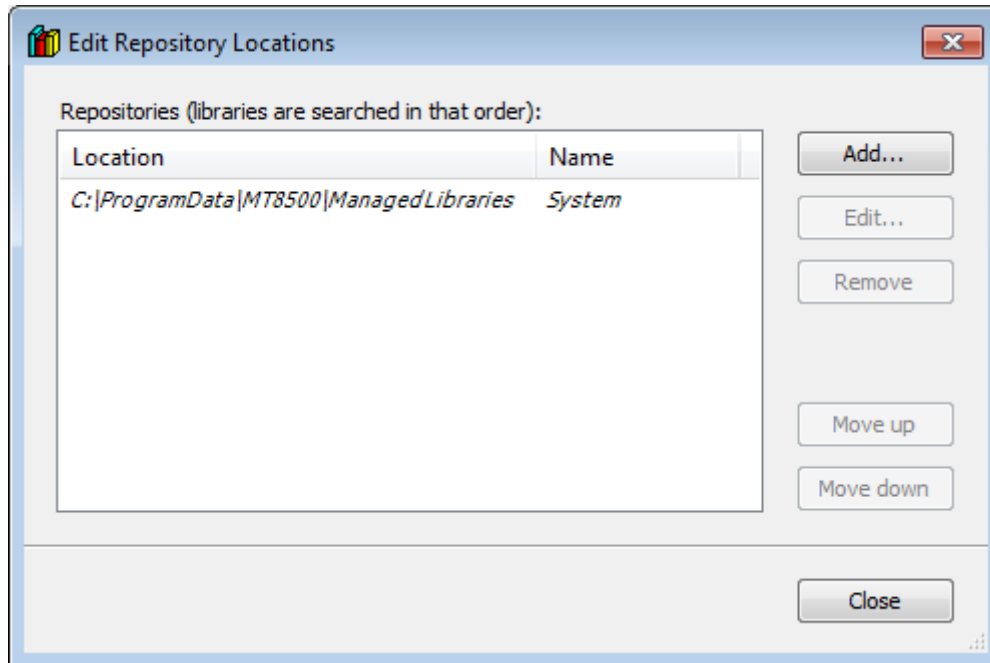


Figure 7-112. Edit Repository Locations

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.

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.

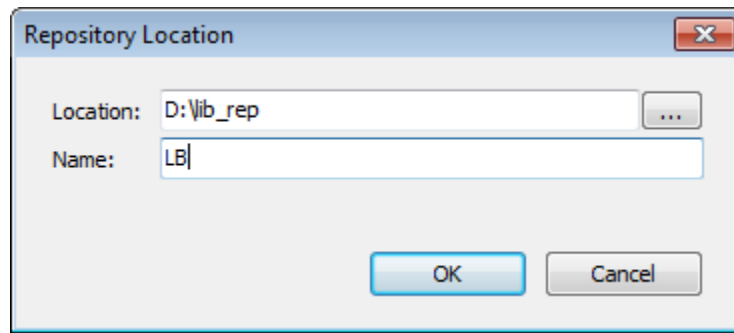


Figure 7-113. Dialog Edit Repository Locations, Add or Edit a Repository

NOTE: Only empty folders can be characterized as a repository. The repository of the "System" is not editable (indicated in italic font).

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.

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.

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.

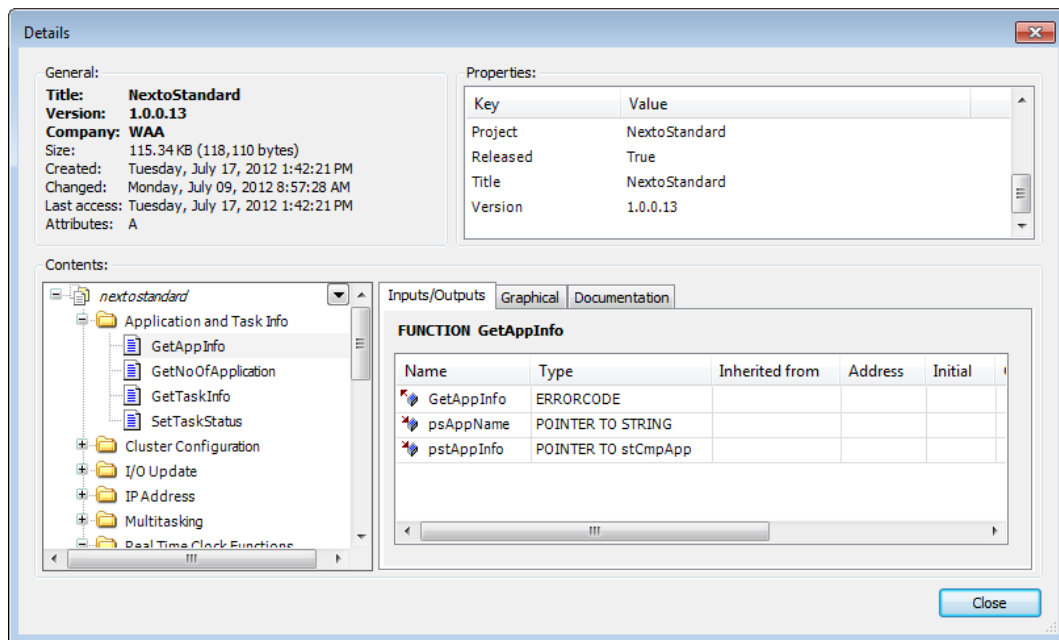


Figure 7-114. Details

The *Dependencies...* button displays the current library dependencies (other libraries included in this) with the following information, title, version and company.

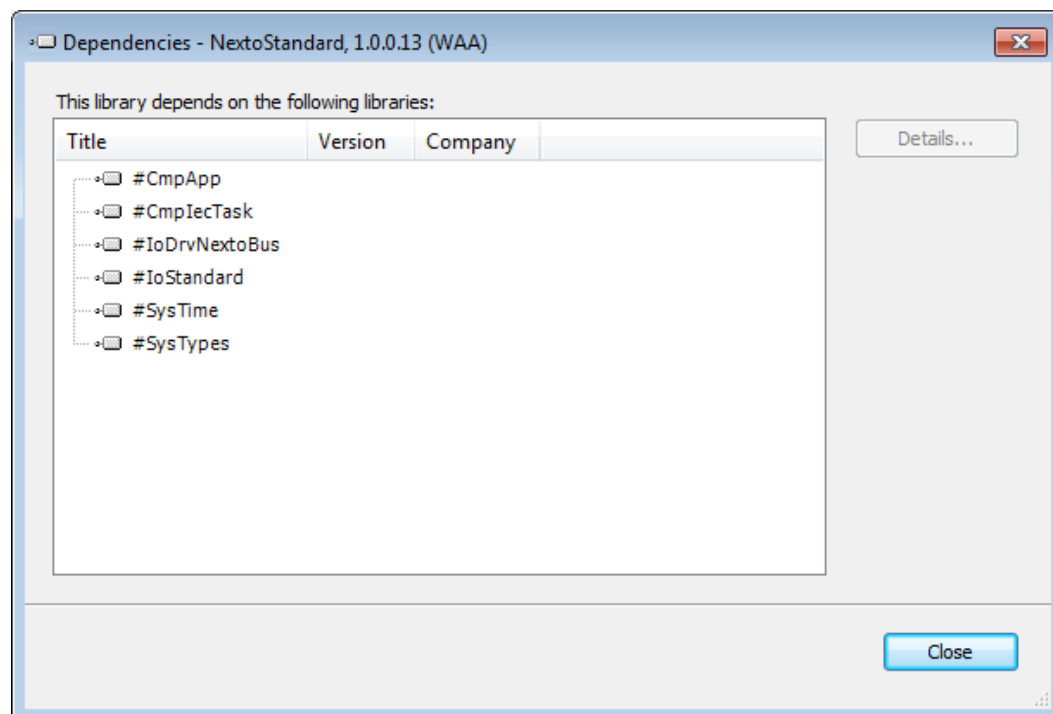


Figure 7-115. Dependencies

Install Library

Symbol: 

Install library is accessed through the Tools menu and opens a dialog for installing libraries. Install library is accessed through the *Tools* menu and opens a dialog for installing libraries.

To perform the installation just select the file and click the *Open* button.

NOTE: The *Install Library* dialog is only available if predefined feature sets chosen by the user are *Standard* or the option *Enable Repository dialog* is disabled. For further information about features, see **Features**.

Device Repository

Symbol: 

Device Repository is accessed through the *Tools* menu and opens the dialog *Device Repository*.

NOTE: The *Device Repository* dialog is only available if predefined feature sets chosen by the user are *Professional* or if the option *Enable Repository dialog* is enabled. For further information about features, see **Features**.

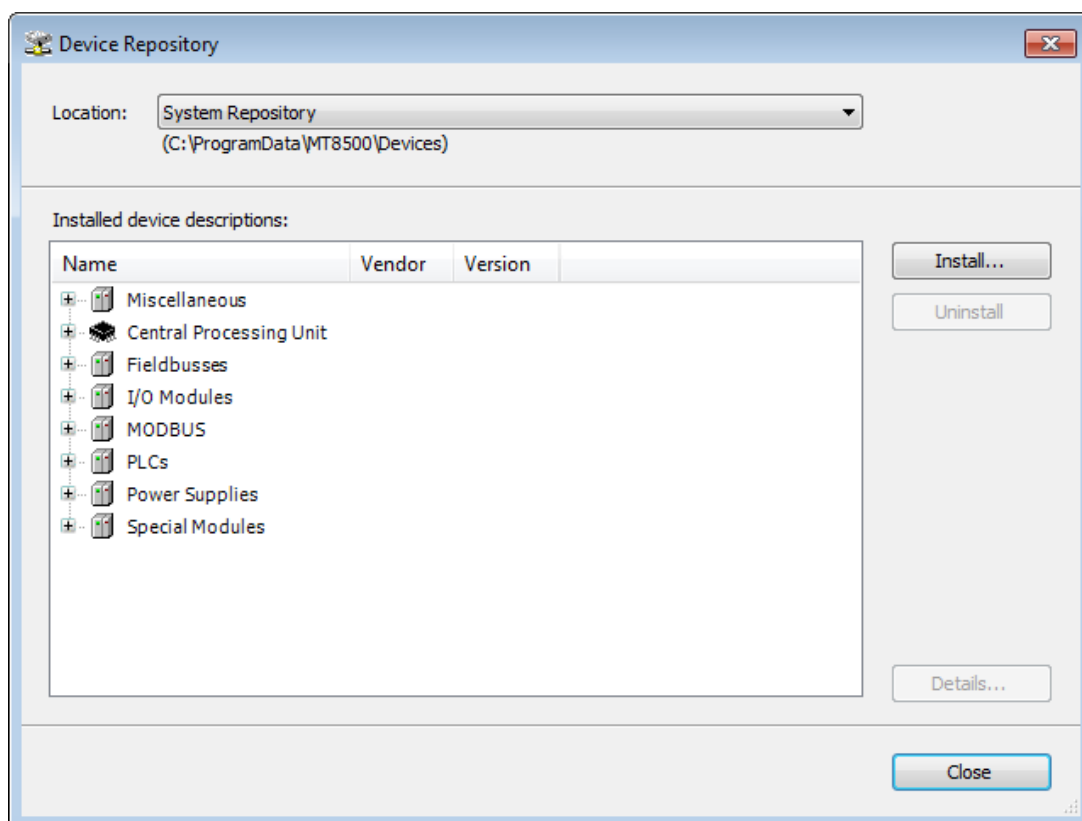


Figure 7-116. Device Repository

A device repository is a database for the devices, which have been installed on the local system for the purpose of being available in the MasterTool IEC XE development system. In the Device Repository, the user can add or remove such device installations.

- *Location:* Device repositories can be available on multiple locations on the system. The selection list offers the currently available locations. By default with the MasterTool IEC XE installation at least the "System Repository" is provided.
- *Installed device descriptions:* The currently installed devices are listed in a tree structure, each showing the Name, Vendor name and Version of the device. The devices tree may be structured by categories like for example "PLCs" and "Miscellaneous" (Figure 7-116). If applicable open or close the tree entries via the plus and minus buttons.

- *Details*: This button opens for the currently selected device the 'Details'-dialog showing additional information as given by the device description file: Device name, Vendor name, Categories, Version, Order Number, Description.
- *Install*: Use this button to get a device installed for being available in the programming system. The dialog *Install Device Description* will open where you can browse your system for the respective device description. For the standard devices the file filter is to be set to “*.devdesc.xml”. But also description files provided by the manufacturer, like for example *.gsd files for Profibus DP modules, can be selected by setting the respective filter.

As soon as you confirm the selection with OK, the dialog will close and the new device will be added to the devices tree in the Device Repository dialog.

NOTE: During installation the device description files and all additional files referenced by that description will be copied to an internal location. Altering the original files will have no further effects on the installed devices. In order for these changes to take effect, the devices will have to be reinstalled. It is considered good practice to change the internal version number of a device description after it has been changed.

ATTENTION:

The internal device repository must never be altered manually. Always use the *Device Repository* dialog to reinstall, add or remove devices.

- *Uninstall*: This command will remove, that is uninstall the currently selected device. It will be removed from the device repository and not be available for use in the programming system any longer.

The current list of installed devices will be offered when you are going to add a device object via *Add Device* or *Add Object*.

Install Device

Symbol: 

Install device is accessed through the *Tools* menu and open the device installation dialog.

To perform the installation just select the file and click the *Open* button.

NOTE: The *Device Install* dialog is only available if predefined feature sets chosen by the user are *Standard* or the option *Enable Repository dialog* is disabled. For further information about features, see **Features**.

Options...

The *Options* dialog, contains sub-dialogs for configuring the behavior and display of the MasterTool IEC XE programming interface.

The current settings, confirmed with OK, will be saved on the local system, overwriting the defaults, which are provided with an installation of the programming system.

Available commands:

- Features
- Load and Save
- International Settings
- CFC Editor
- Declaration Editor
- SFC editor

- Text Editor
- FBD, LD and IL editor
- Syntax Highlighting
- SFC
- SmartCoding

Features

This subdialog in the *Options* dialog provides the possibility to adapt the selection of available features in the currently used programming system environment. In the upper part you can switch on and off the particular features, in the lower part you can switch between the predefined sets on a whole.

NOTE: Some settings restrict the selection of object types that can be newly created in a project. However, if the user opens a project that already contains objects of currently not insertable types, these nevertheless will be visible and fully functional.

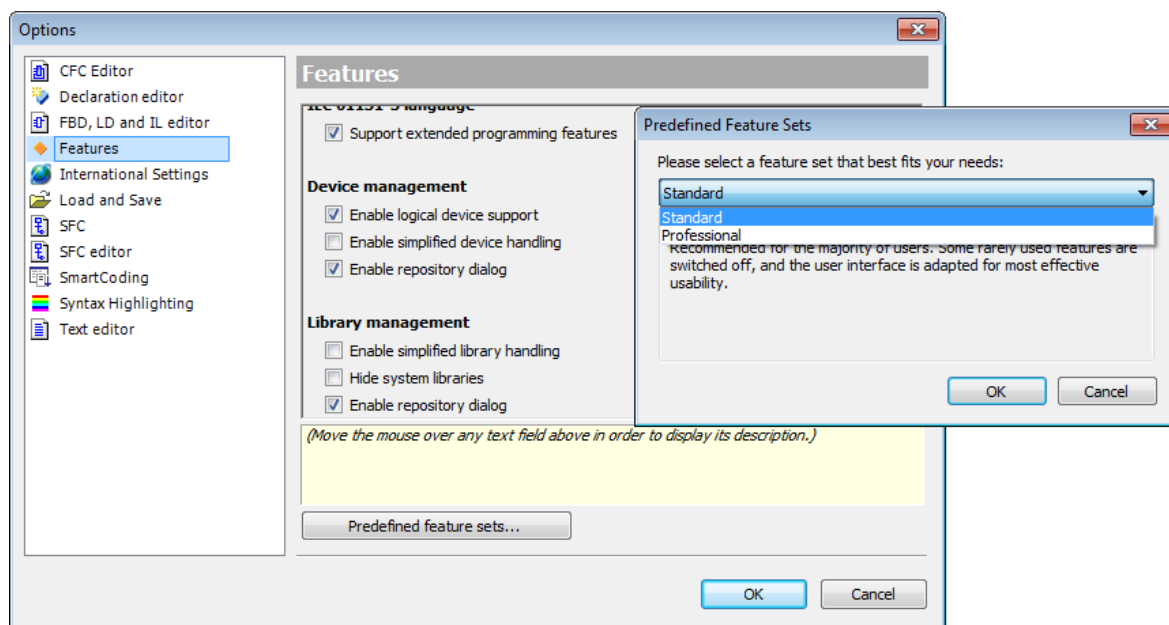


Figure 7-117. Options Dialog, Features

Predefined Features Sets

Currently there are two predefined sets of features, which can be exchanged via button *Predefined feature sets*.

Standard: Some rarely used features are switched off and the user interface is adapted for most effective usability. Recommended for the majority of users.

Professional: All features are available and the user interface is more complex in order to unveil all possibilities of the system.

Particular Features

Alternatively, each of the features, which are only available with the *Professional* set, can be activated (on) or deactivated (off) in particular. For this purpose add or remove a checkmark in the respective option box. The concerned features belong to the categories described in Table 7-3. In the last two columns each see the default setting for the Standard and Professional set.

Category	Description	Standard	Professional
Smart Coding			
Show system symbols	If this feature is activated symbols of the system library will be available on the input assistant.	Off	On
IEC 61131-3 Languages			
Support extended programming features	Only if this feature is activated, the following object types will appear in the "Add object" context menu: - Method - Property Furthermore, when creating a new POU, the wizard will offer to extend FBs or implement interfaces. The new tabular declaration editor also will offer to extend FBs, implement interfaces, and decorate the entire POU or single declarations with compiler pragmas. The input assistant will offer declaration keywords that are used together with object oriented programming (EXTENDS, etc..).	Off	On
Device Management			
Enable logical device support	If activated, the user enables the access to logic devices.	On	Off
Enable simplified device handling	If activated, the user will only see the newest device description version for a particular device; the version itself is only displayed at some places for informational purposes).	On	Off
Enable repository dialog	If activated, the menu command Tools / Device Repository... will be available. Otherwise, the command 'Tools / Install Device...' will appear instead, providing access to the Install Device dialog. Another possibility is to install new device descriptions using the package mechanism.	Off	On
Library Management			
Enable simplified library handling	If activated, the following simplifications will apply: - The user will not be able to save a project as a library. - The user will not be able to install a project to the library repository. - The user will not be able to open library projects for editing. - When adding a library in the Library Manager, there is no possibility to add placeholder references. - When editing the properties of library reference, only the namespace can be adapted. - In the Library Manager list itself, the version column is not displayed. - When adding a library in the Library Manager, only the newest version per particular library is displayed; the version information itself is omitted.	On	Off
Hide system libraries	If activated, system libraries ("gray libraries") will not be displayed in the Library manager. Exception: if such a library could not be loaded, it will	On	Off

	be displayed so that the user can remove the reference manually.		
Enable repository dialog	If activated, the menu command Tools / Library Repository... will be available. Otherwise, the command 'Tools / Install Library...' will appear instead, providing access to the Select Library dialog. Another possibility is to install new libraries using the package mechanism.	Off	On

Table 7-3. Available Features (Professional Category)

Load and Save

This sub-dialog of the *Options* dialog allows settings concerning the behavior of MasterTool IEC XE at loading and saving a project.

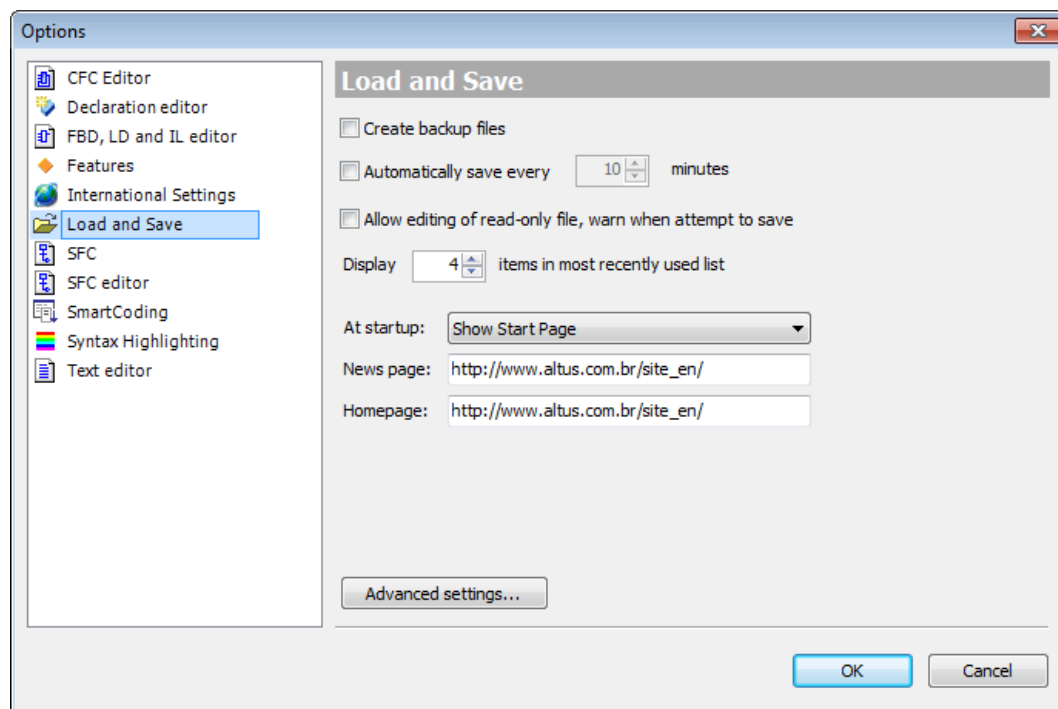


Figure 7-118. Load and Save Options Dialog

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, the *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.

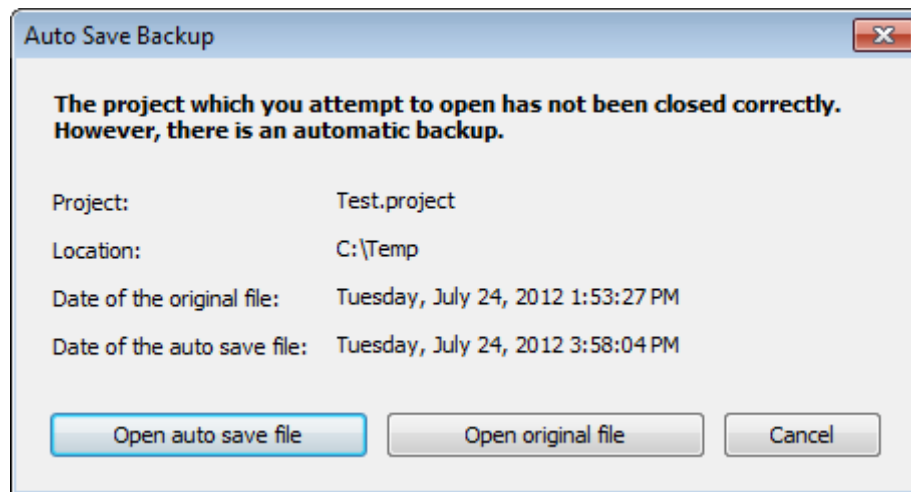


Figure 7-119. Auto Save Backup Dialog

On *Open auto save file* option the file <projectname>.autosave will be opened and marked as modified.

On *Open original file* option, the last version of the project saved by the user will be opened.

On *Allow editing of read-only file* option of the *Load and Save* dialog, warn when attempt to save. You can edit a project, which is restricted to read-only access, but nevertheless an alert (warning) will be generated when you are going to save the file.

In *Display ... items in most recently used list* is defined how many of the recently opened projects should be displayed in the *Recent Projects* list (*File* menu).

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*.

News page is URL of the page, which is displayed in the *News* area of the *Start Page*.

Homepage is URL of the page, which will be displayed on command *Altus Home Page* (*Help* menu).

The *Advanced Settings...* button for opening the same-named dialog:

- *Project Compression*: See the dialog for further information on the three possible compression levels. Per default the least compression is set.
- *Load behavior*: If this option is activated (default and recommended) the loading of libraries and compile information will be done in the background, while you already can start working on the project.

International Settings

This sub-dialog of the *Options* dialog allows the following settings concerning the language used in the user interface and help system.

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.

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.

CFC Editor

This sub-dialog of the *Options* dialog provides settings concerning the editing in a CFC (Continuous Function Chart) editor.

The settings will immediately be applied to the currently opened editor views as soon as they have been confirmed (and the dialog closed) with *OK*.

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.

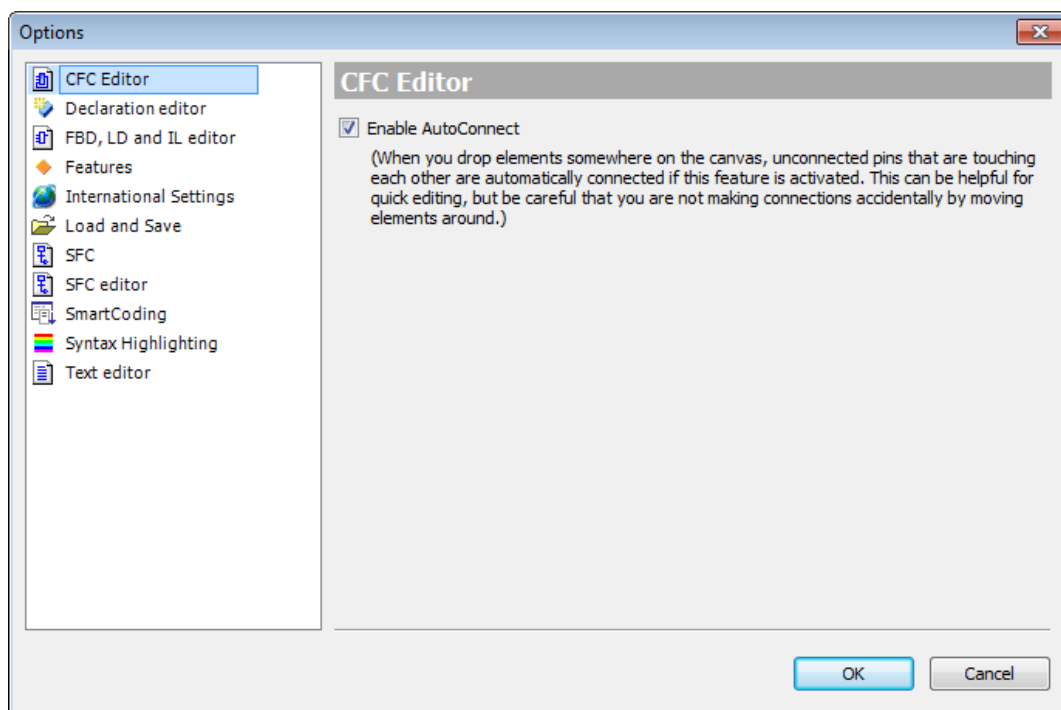


Figure 7-120. Enable AutoConnect

Declaration Editor

This sub-dialog of the *Options* dialog contains some default settings on the appearance of the declaration editor.

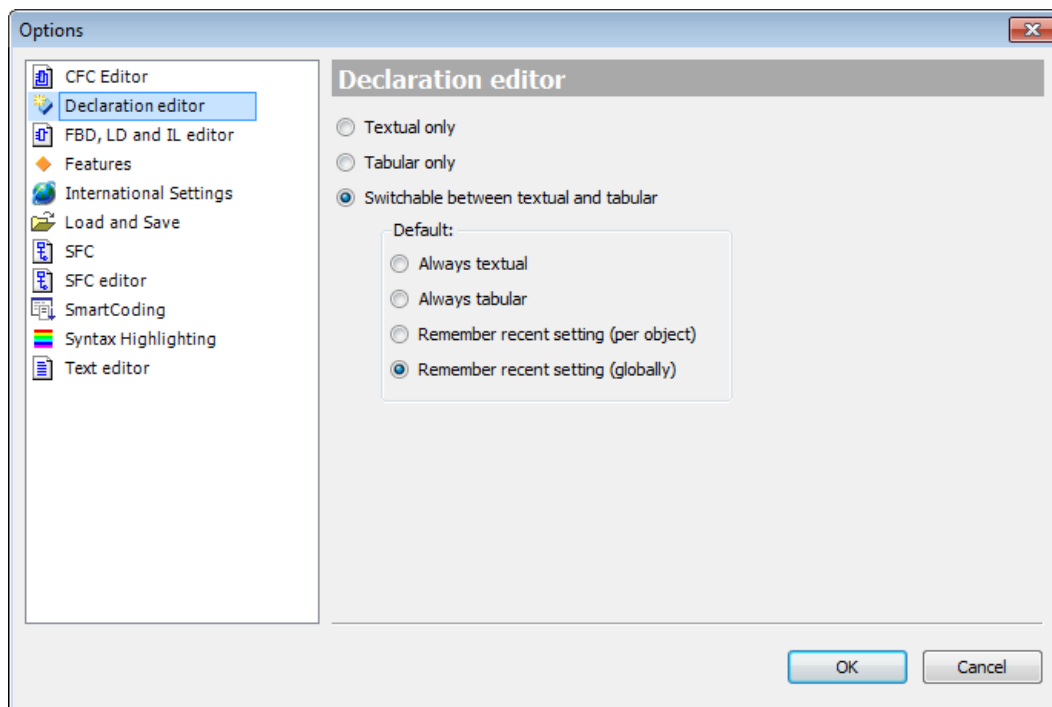


Figure 7-121. Declaration Editor Options

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).

Editor SFC

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*.

Layout

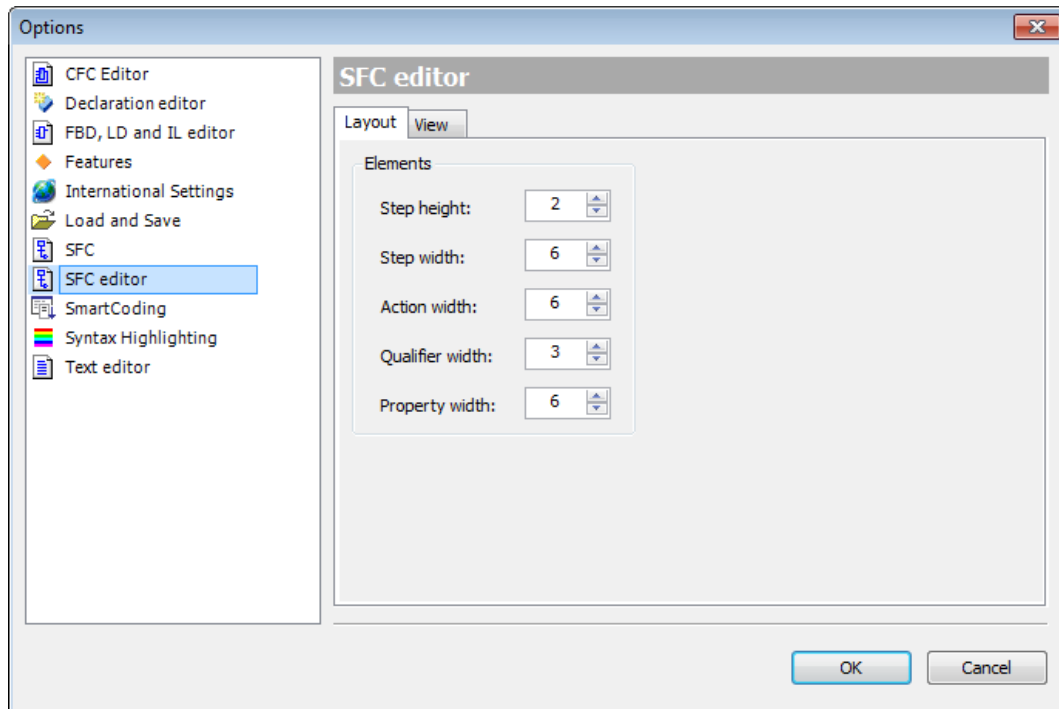


Figure 7-122. SFC Editor Options Dialog, Category Layout

The settings are to be made in “grid units”. One grid unit is equal to font size currently set in the text editor options.

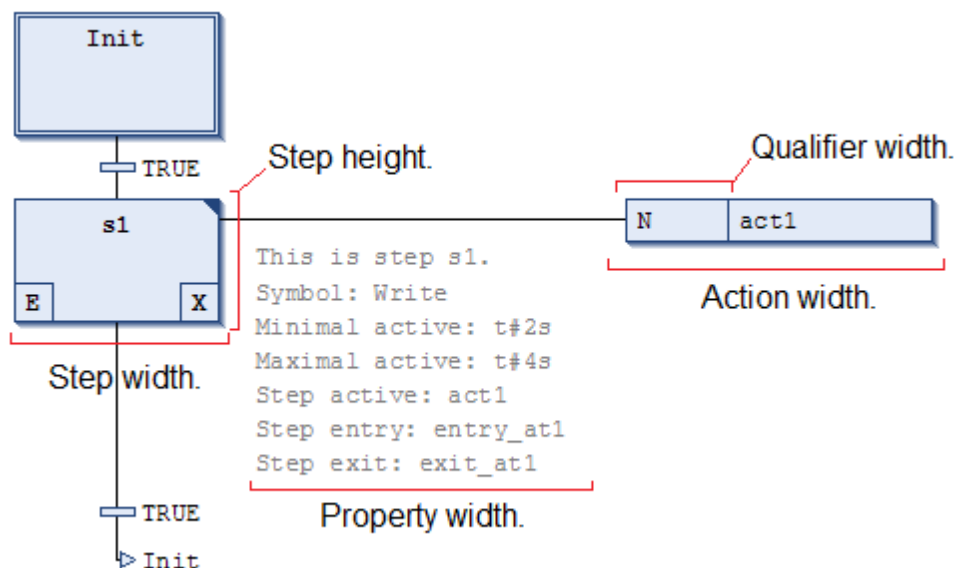


Figure 7-123. Layout Options

- *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.

View

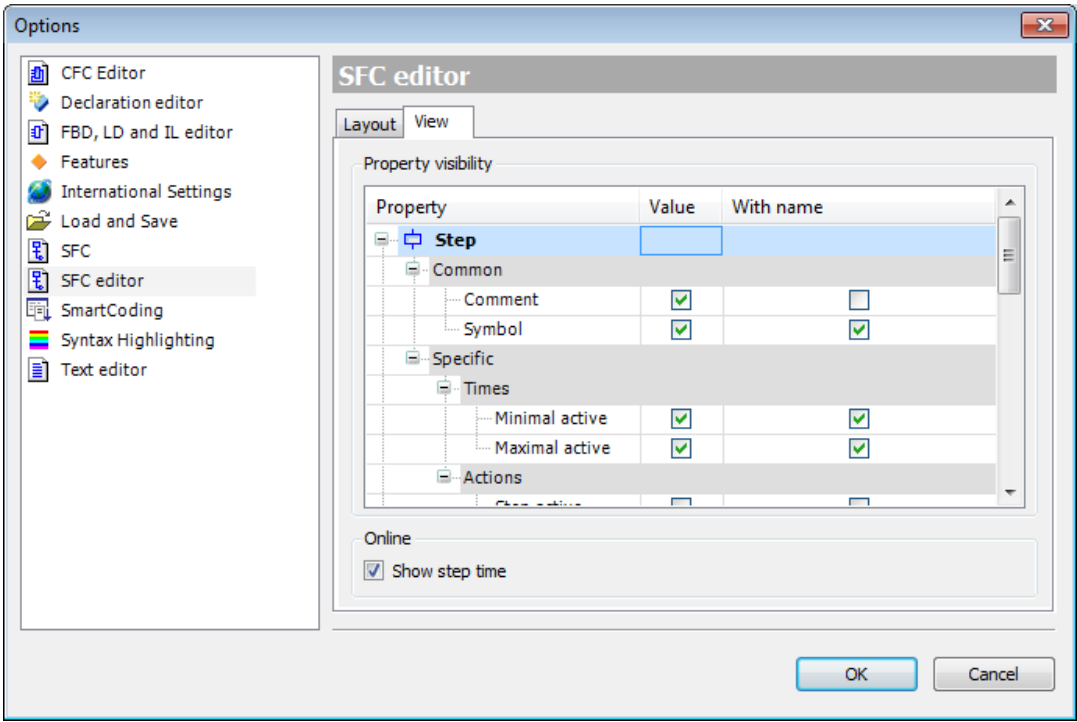


Figure 7-124. SFC Editor Options Dialog, Category View

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.

Properties of the elements.

Properties	
Filter	Sort by Sort order
Property	Value
Common	
Name	s1
Comment	This is step s1.
Symbol	Write
Specific	
Initial step	☐
Times	
Minimal active	t#2s
Maximal active	t#4s
Actions	
Step active	act1
Step entry	entry_at1
Step exit	exit_at1

SFC editor options.

SFC editor		
Layout View		
Property visibility		
Property	Value	With name
Step		
Common		
Comment	☒	☐
Symbol	☒	☒
Specific		
Times		
Minimal active	☒	☒
Maximal active	☒	☒
Actions		
Step active	☒	☒

SFC editor.

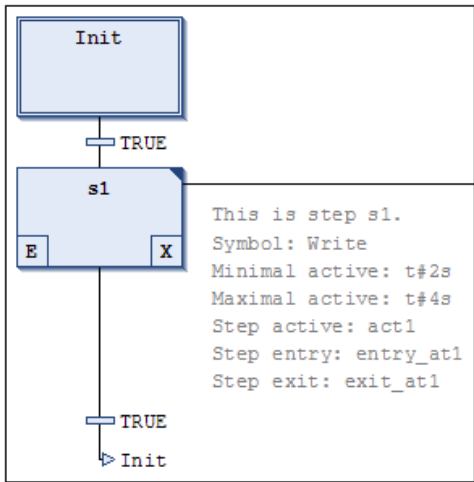


Figure 7-125. Property Visibility Example

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.

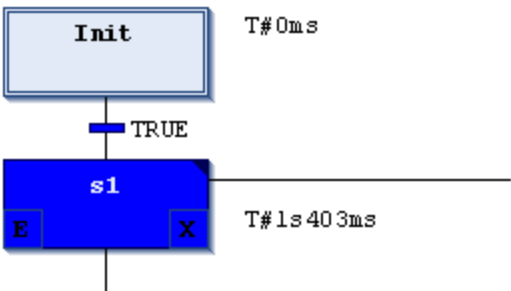


Figure 7-126. Step Time Displayed in Online Mode

Text Editor

This sub-dialog of the *Options* dialog allows settings concerning the editing in a text editor.

Editing

Definition of undo, tabulator, indenting, folding etc. in the editing area of the text editor.

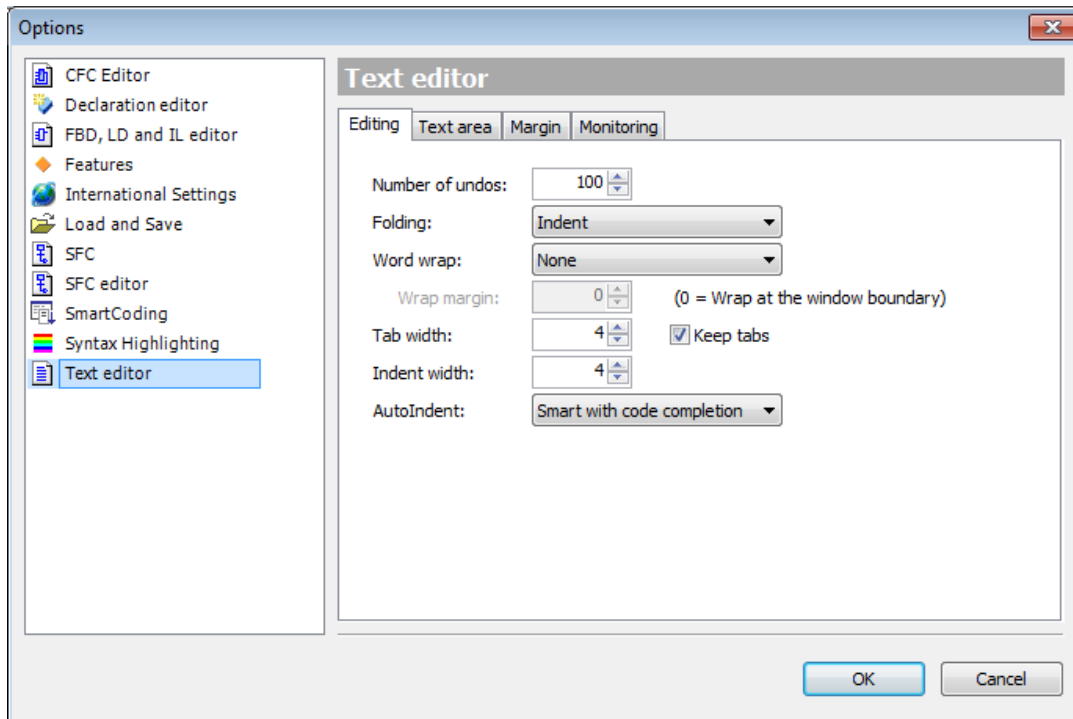


Figure 7-127. Options Dialog, Category Text Editor, Editing

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)

Examples: 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).

Opened and closed folds, examples:

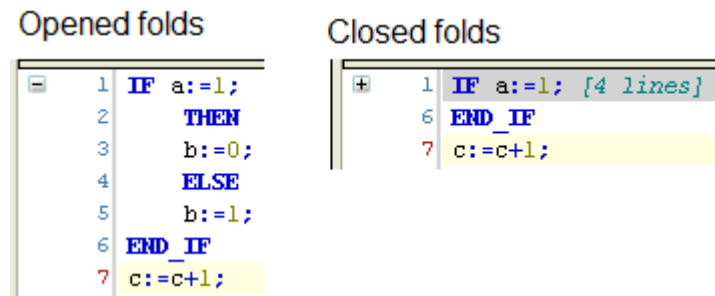


Figure 7-128. Opened and Closed Folds

- *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.

Examples for defining a folding section:

```

1  IF b THEN
2  /// {{{
3  IF a THEN
4      c:=TRUE;
5  END_IF
6  /// }}}
7  END_IF

2  /// commentcomment {{{ commentcomment
3  IF a THEN
4      c:=TRUE;
5  END_IF
6  /// }}}
7  END_IF

```

Figure 7-129. Explicit Folds Defined by Special Comments

- **Word Wrap:**
 - *None*: The line can be filled endlessly.
 - *Soft*: 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.

Text Area

Color and font definitions for the editing area of the text editor.

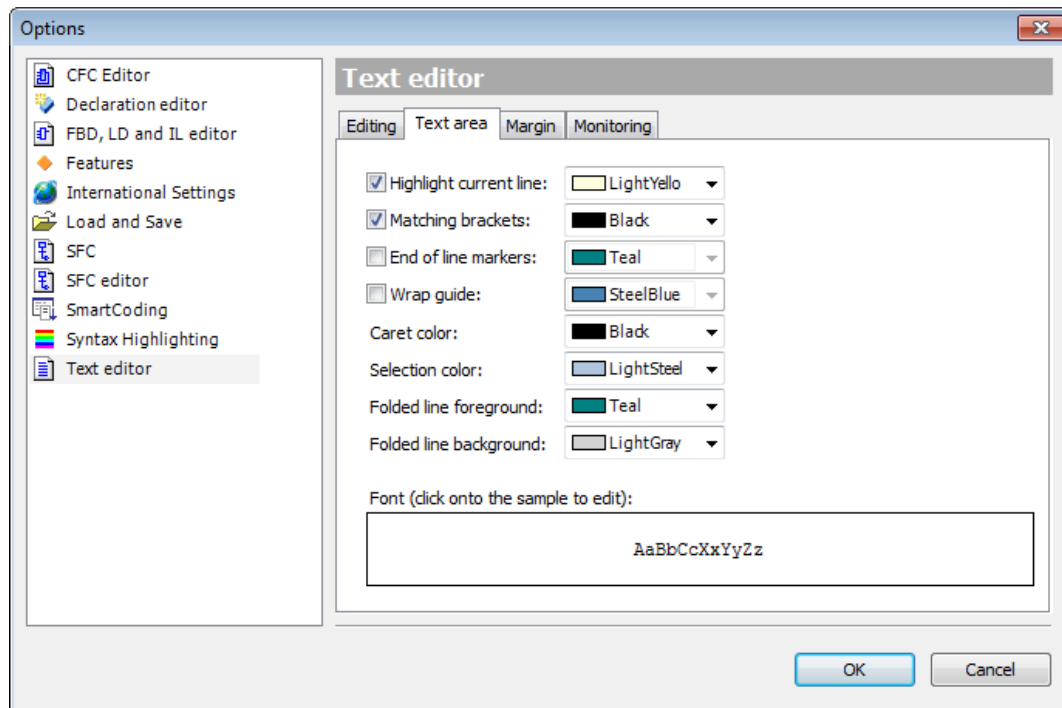


Figure 7-130. Options Dialog, Category Text Editor, Text Area

You can activate the following options in order to have some visual assistance during editing:

Field	Meaning
Highlight current line	The line where the cursor is placed is highlighted with the color defined in the selection list.
Matching brackets	If the cursor is placed before or after a bracket "(" or ")" in a code line, the corresponding closing resp. opening bracket will be indicated by a rectangle, which is colored as defined in the color selection field. Also, if the cursor is placed in the first or last line of a bracket scope*, the corresponding line at the end and start of this scope will be indicated also by a rectangle of this color. * A bracket scope contains all lines between corresponding keywords (for example IF and END_IF, see also tab 'Margin')
End of line markers	The end of each editor line will be marked by a small dash of the defined color, which is placed behind the last position, which is occupied, by a character or a white-space.
Wrap guide	If soft or hard word wrap is activated (see above, tab Editing), the defined wrap margin will be indicated by a vertical line in the defined color.
Caret color	Color of the cursor caret:
Selection color	The currently selected area will be highlighted with this color.

Folded line foreground	The header of a closed, folded section of code lines will be written in this color.
Folded line background	The header of a closed, folded section of code lines will be highlighted in this color.
Font	Choose the font to be used in the text editor. By a mouse-click on the sample field you get the standard dialog for setting the font.

Table 7-4. Visual Help to Edition

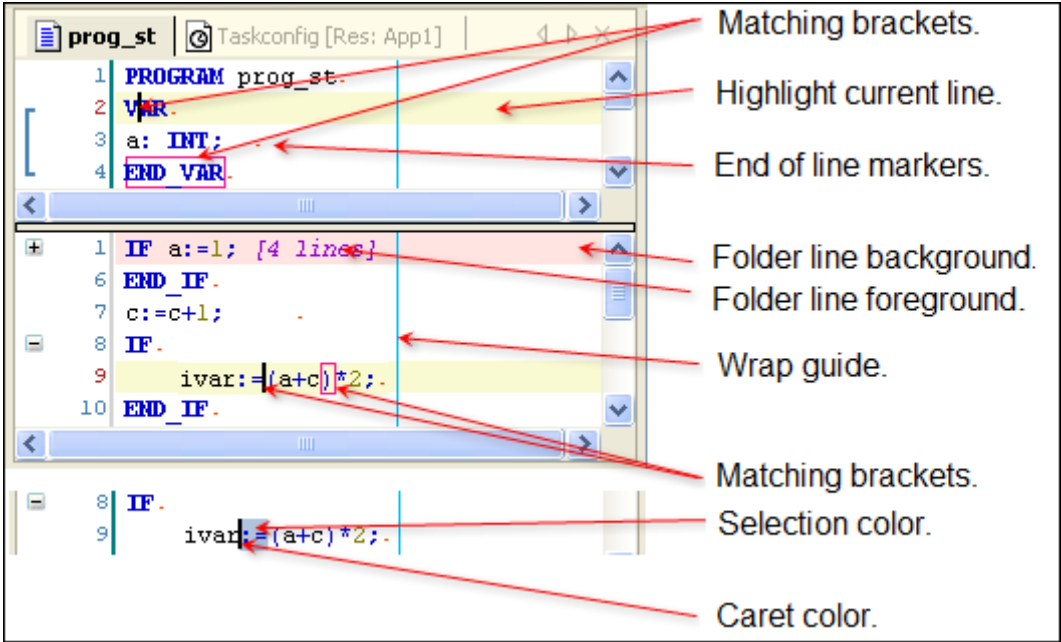


Figure 7-131. Examples for the Settings in the Text Area Dialog

Margin

Definition of colors, font, mouse-click definitions concerning the area left to the editing area of the text editor window.

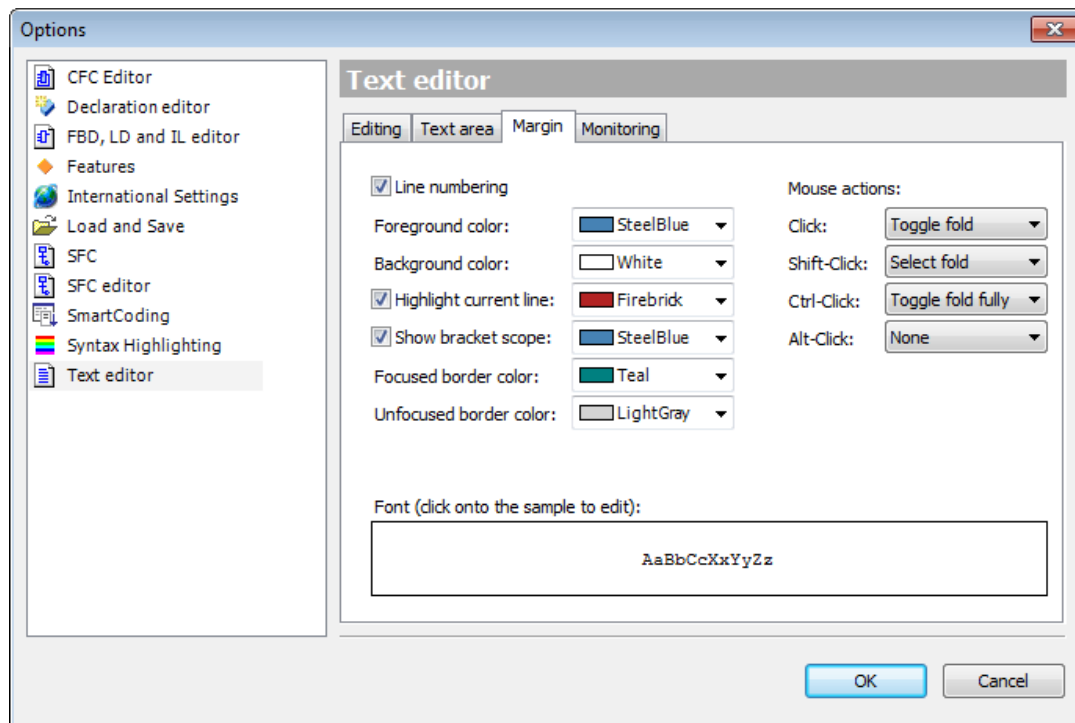


Figure 7-132. Options Dialog, Category Text Editor, Margin

The following options concern the left margin of the editor window, which is separated by a vertical line to the editing area:

Field	Meaning
Line numbering	Line numbers will be displayed in the declaration and in the implementation part of the window, each starting with 1 in the first line.
Foreground color	Color of the line numbers.
Background color	Color of the margin background.
Highlight current line	Color of the line number of the currently edited line.
Show bracket scope	A bracket scope contains lines between corresponding keywords, like e.g. between IF and END_IF. You can activate this bracket scope function by option 'Matching brackets' in category 'Text area'. If the cursor then is placed before, after or within a corresponding keyword, the bracket scope will be indicated by a bracket in the margin area.
Focused border color	Color of the vertical separator line on the right side of the margin area in the currently active part of the editor.
Unfocused border color	Color of the vertical separator line on the right side of the margin area in the currently not active part of the editor.
Mouse actions	Define which of the following actions should be performed when you perform the associated type of mouse-click on the plus- resp. minus sign before the header line of a fold: - None: No action. - Select fold: The lines of the fold area will be selected. - Toggle fold: The first level of the fold will be opened and closed. - Toggle fold fully: All levels of a fold will be opened and closed. Types of mouse clicks: Click, Shift-Click, Ctrl-Click, Alt-Click.

Table 7-5. Left Margin Options

Monitoring

Activating/Deactivating and configuration of the display of inline monitoring field.

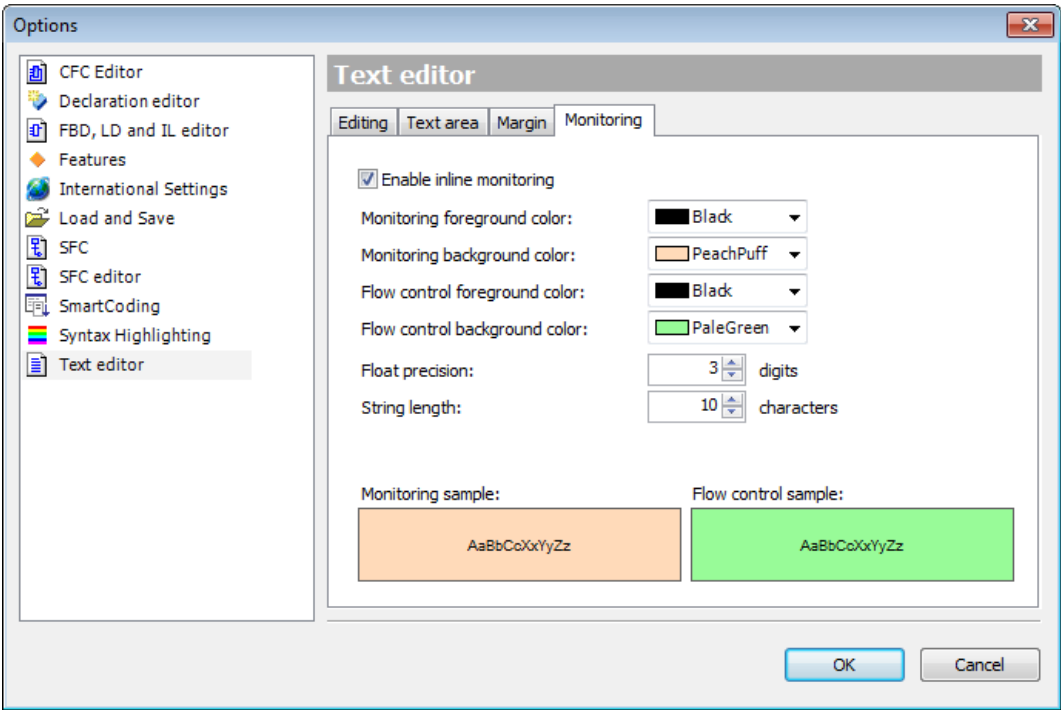


Figure 7-133. Options Dialog, Category Monitoring

Button	Meaning
Enable inline monitoring	Activate/deactivate inline monitoring.
Monitoring foreground color	From the selection list choose a color for the writing in the monitoring fields.
Monitoring background color	From the selection list choose a color for the background of the monitoring fields
Flow control foreground color	From the selection list, choose a foreground color of flow control fields.
Flow control background color	From the selection list, choose a color for the background of flow control fields.
Float precision	Define how much decimal places (digits) should be maximum displayed in case of floating values.
String length	Define the number of characters maximum displayed in.
Sample	Preview of the currently defined monitoring settings.

Table 7-6. Monitoring Fields

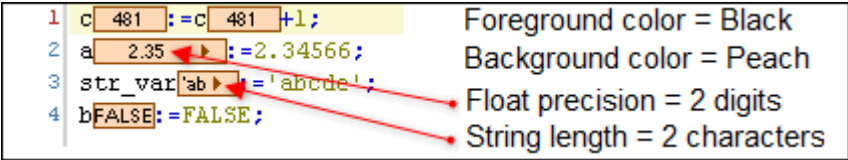


Figure 7-134. Example of Inline Monitoring in ST Editor

FBD, LD and IL Editor

This dialog of *Options* enables user to perform settings for editing in editors FBD, LD and IL.

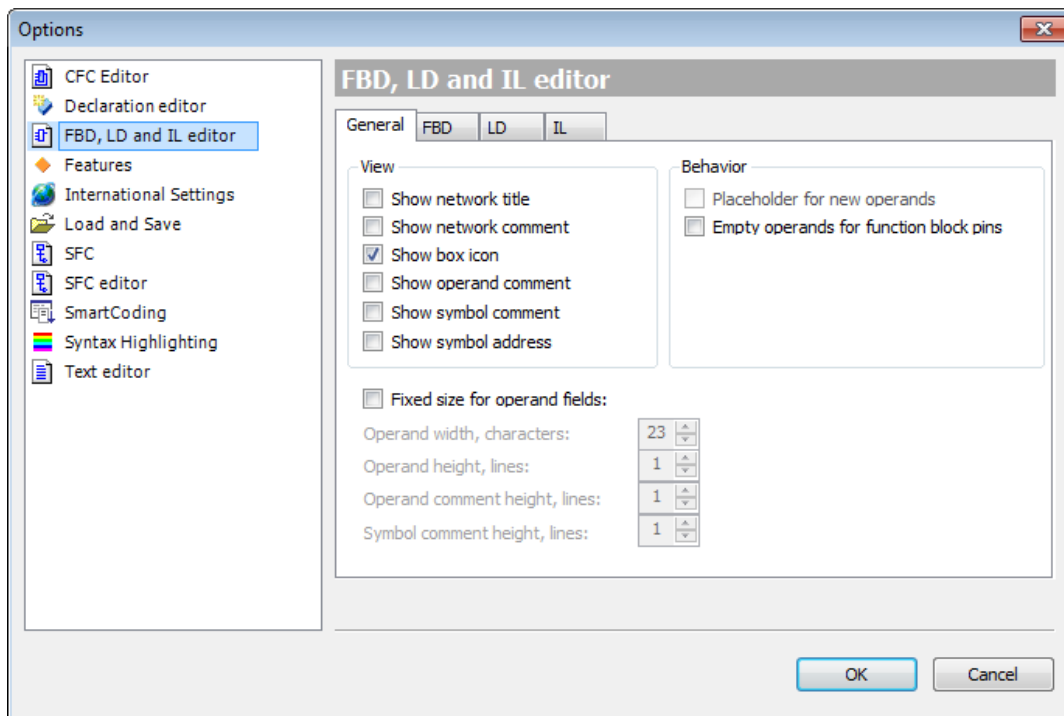


Figure 7-135. FBD, LD and IL Editors Viewing Options

- *Show network title*: the title of the network - if set - is displayed in the upper-left corner of a network.

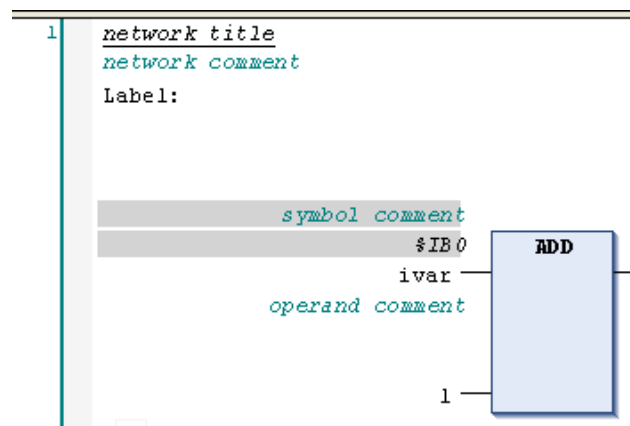


Figure 7-136. Components in the Editor's View

- *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.

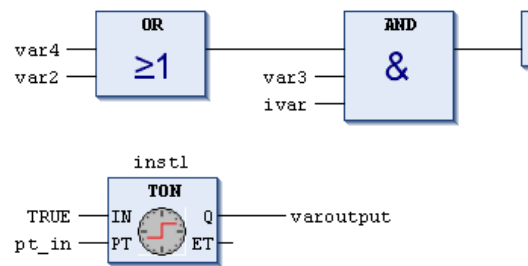


Figure 7-137. Boxes with icons for standard operators on a FBD network

- *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.
- *Fixed operand size*: 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.
- *Placeholder for new operands*: not available.
- *Empty operands for function blocks pins*: with this option enabled empty operands are allowed for pins of function blocks.
- *Connect boxes with straight (FBD Tab)*: 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). Example:

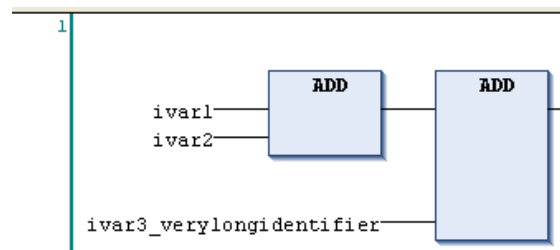


Figure 7-138. Enabled option

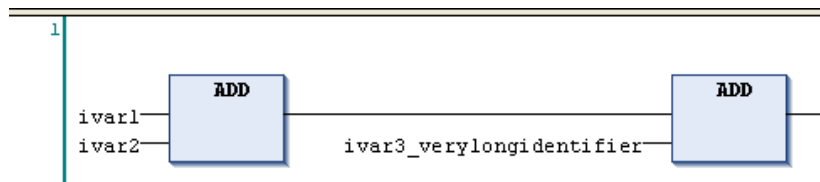


Figure 7-139. Disabled option

- *Networks with linebreaks (FBD and LD Tabs)*: 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.
- *Default network content (FBD, LD and IL Tabs)*: defines what content is displayed in the editor when inserting a new network (empty, assigned or empty box).
- *After inserting selection (FBD, LD and IL Tabs)*: this option defines which element is selected after a new network is entered (Network or Element).

Syntax Highlighting

This sub-dialog of the *Options* dialog allows the definition of colors and font style for the particular text items (for example operands, pragmas, comments etc.) in an editor window.

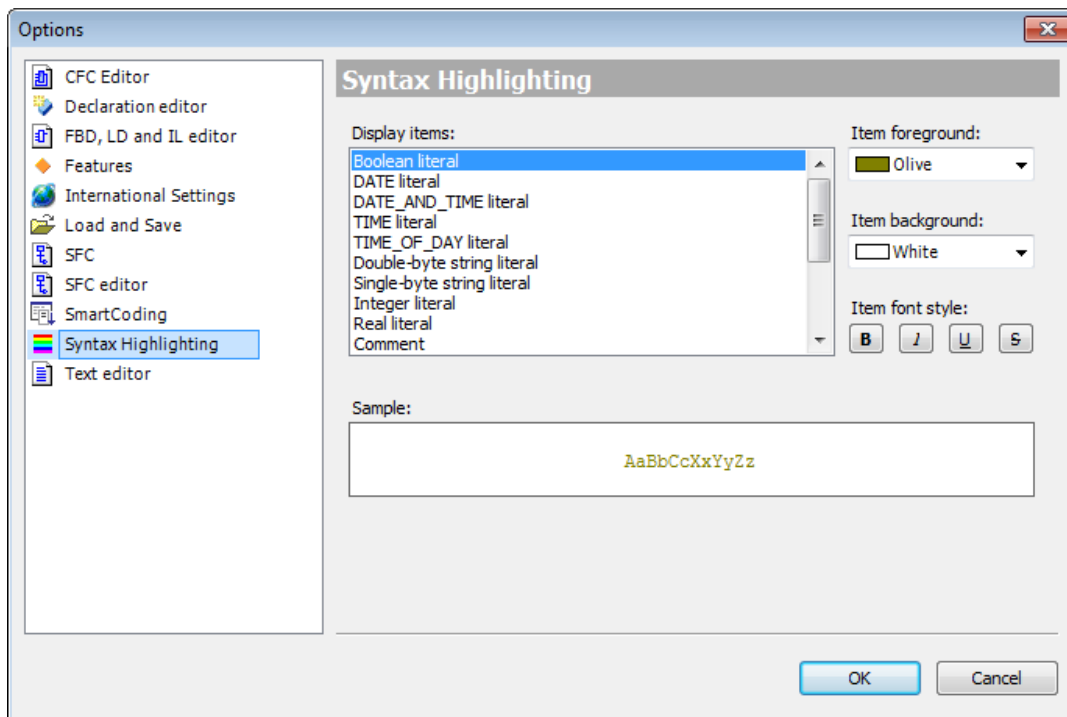


Figure 7-140. Options Dialog, Category Syntax Highlighting

Define here which color and font style should be used for displaying the particular items in the text editor window. Select an item type from the *Display Items* selection list, then select the desired colors for the Item foreground and Item background) and define the Item font style by a mouse-click on one or several of the respective buttons for bold (B), italic (I), underlined (U) and strike through (S).

A sample view of the current settings will be displayed in the *Sample* window.

SFC

This sub-dialog of the *Options* dialog provides the possibility to set default settings for SFC objects. Each new SFC object will get these defaults in its properties.

NOTE: Notice that any modifications in the default settings will not be applied to existing SFC objects until you use the *Set defaults* button in the *SFC Settings (Properties window)* of the particular object. The object properties have higher priority than the defaults.

The pure editor options that is settings for layout and view are handled in a separate *SFC Editor* dialog.

Flags

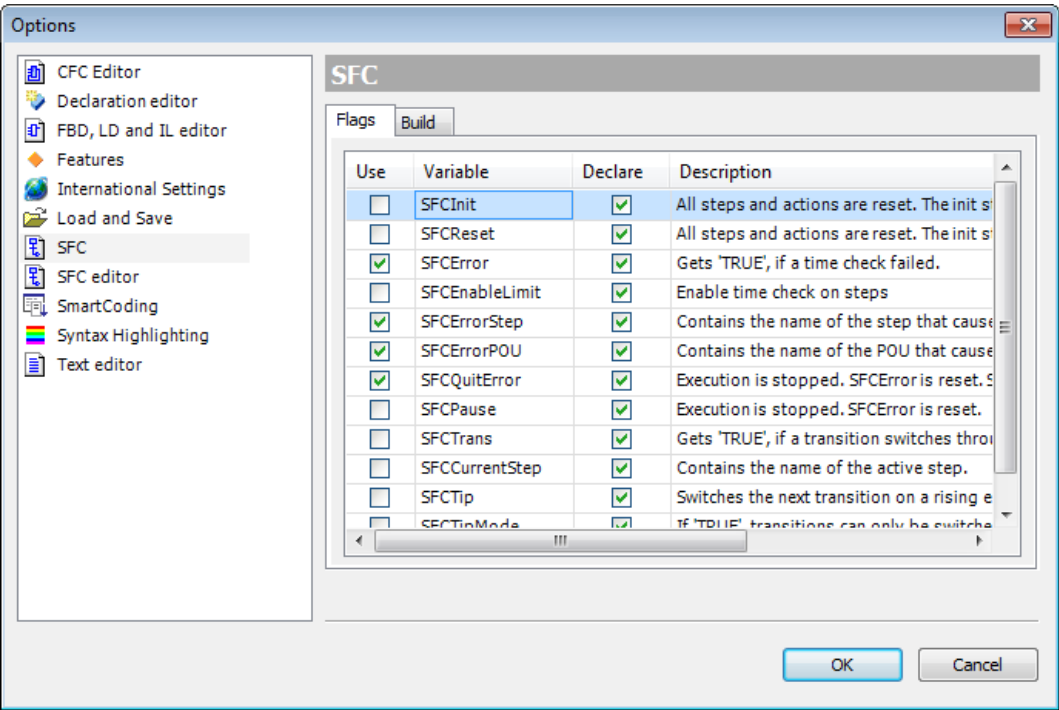


Figure 7-141. SFC Options Dialog, Flags

All possible flags, that is implicitly created variables for watching and controlling the processing of a SFC, are listed in this dialog. A short description is added, but you might see **Implicit Variables – SFC Flags** for further details (IEC 61131 Programming Manual).

By a mouse-click in the corresponding checkbox you can activate the automatic declaration (*Declare*) and use (*Use*) of a flag. These settings will be applied as defaults to new SFC objects. If *Declare* is activated, but *Use* is deactivated for a flag, the declaration will be done, but the flag will be without any effect during processing.

Notice that an automatically declared flag variable will only be visible in online mode in the declaration part of the SFC Editor.

A timeout has been detected in step “s1” in SFC object “POU” by flag SFCError

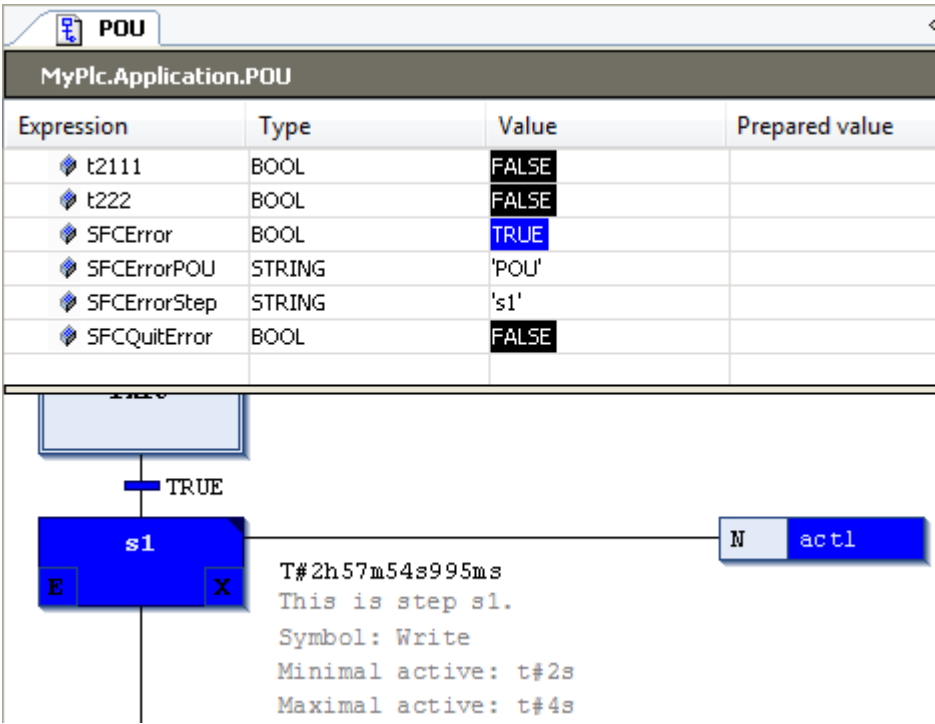


Figure 7-142. Example of Some SFC Error Flags in Online Mode of the Editor

Build

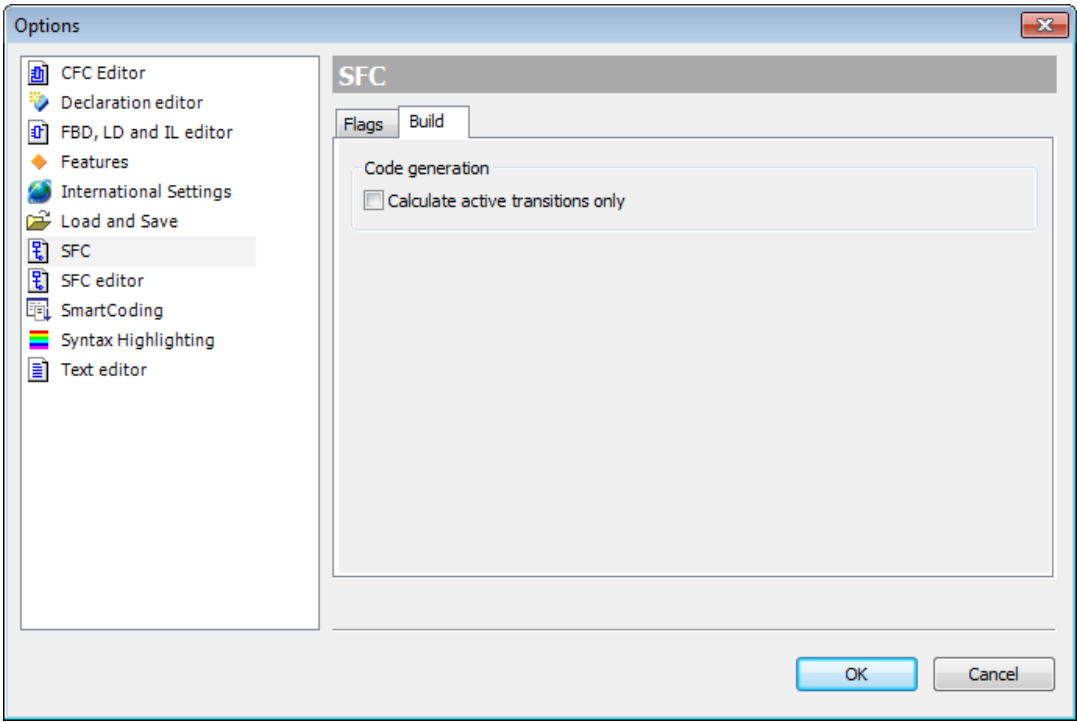


Figure 7-143. SFC Options Dialog, Build

- *Calculate active transitions only:* If this option is activated, by default code will be generated only for the currently active transitions.

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*.

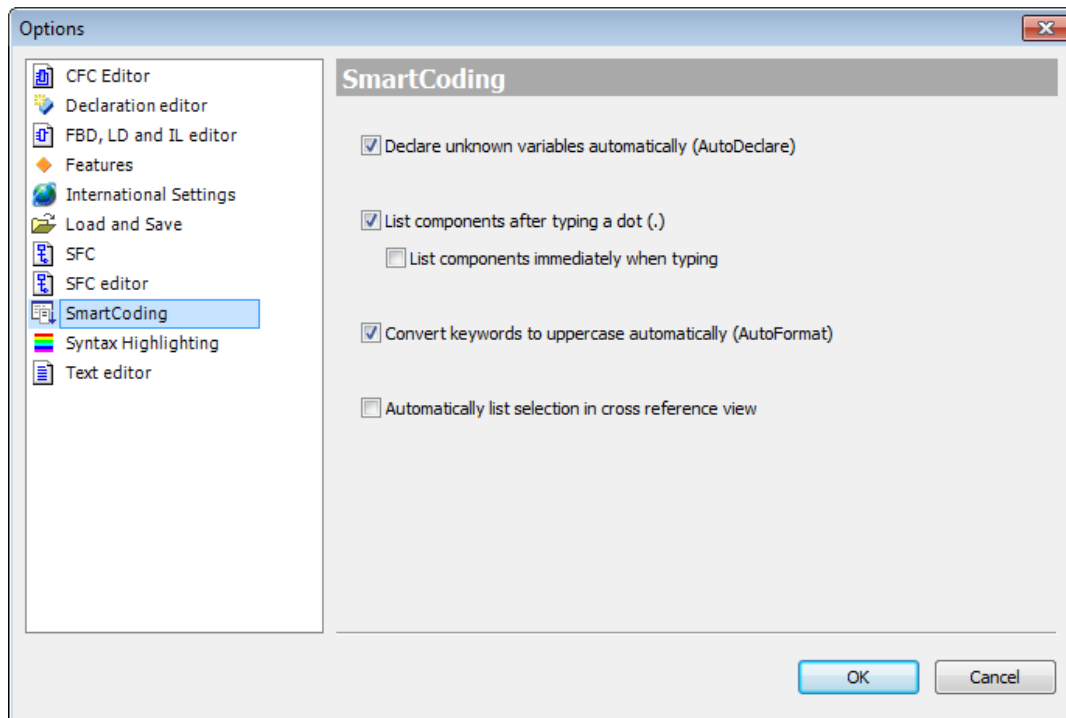


Figure 7-144. SmartCoding Options Dialog

- *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.
- *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.
- *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 "iVar: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.

Window Menu

The frame provides commands for command category *Window* that can be used to handle the user interface windows.

Available commands:

- Next Editor
- Previous Editor
- Close 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...

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 next one. This is the window represented with the tab to the right of the currently active window.

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.

Close Editor

Default Shortcut: <CTRL>+<F4>

This command closes the currently active editor window (not applicable to the *Configuration (Bus)*).

Close All Editors

Symbol: 

This command closes all currently opened editor windows (not applicable to the *Configuration (Bus)*).

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.

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.

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.

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.

Dock

This command can be used to dock a window, which previously has been undocked by a *Float* command.

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.

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.

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.

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.

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*.

To close one or several windows, select those entries and use button *Close window(s)*. Multiple selection is possible.

Close will close the dialog (not applicable to the *Configuration (Bus)*).

Help Menu

The *Help* menu provides functions to get links and online help on MasterTool IEC XE.

Available commands:

- Contents
- Index
- Search
- Contact support
- Update Software License...
- Documentation
- Altus Home Page...
- About...

Further commands might be available, indicating links on an Internet page.

Contents

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.

Index

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.

Search

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.

Contact Support

This command opens the Altus web site, directly on the technical support page.

Update Software License

This command opens a dialog box where you can change the company name, serial number and software key.

Documentation

It allows accessing the MasterTool IEC XE documentation.

Available commands:

- Technical Specifications
- Programming Manual
- User Manual

Technical Specifications

This command opens the default web browser to the Technical Specifications document on its most current version available on Altus website.

Programming Manual

This command opens the default web browser with the Programming Manual on its current version available on Altus website.

User Manual

This command opens the default web browser with the User Manual on its current version available on Altus website.

Altus Home Page

This command opens the Altus website.

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.

Bus Editor Menu

Contains commands used for bus editor customization, it is displayed in the menu bar shape and context menu and is available when the editor is active.

Available commands:

- Zoom out
- Zoom in
- Rulers
- Ports
- Grid

Zoom out

Decreases the zoom level in the editor bus screen.

Zoom in

Increases the zoom level in the editor bus screen.

Rulers

Adds rulers on the left side and at the top of the editor bus window.

Ports

Displays in the graphical editor the connection ports related to modules, racks and cables. They serve as guidance to the user of the possible connection points on the graphic objects.

Grid

Adds a grid as a background in bus editor.

Declaration

The “Declaration” category provides commands for working in the tabular declaration editor.

Tabular Declaration Editor - Declaration

It provides commands for working in the tabular declaration editor. The commands are available in the context menu and in the toolbar of the editor.

Available commands:

- Edit Declaration Header
- Insert
- Move Down
- Move Up

Edit Declaration Header

This command opens the editor for the declaration header. For detailed information please see the description of the **Tabular Declaration Editor**.

Insert

This command adds a new line for a variable declaration. For detailed information please see the description of the **Tabular Declaration Editor**.

Move Down

This command moves the currently selected declaration line one down. For detailed information please see the description of the **Tabular Declaration Editor**.

Move Up

This command moves the currently selected declaration line one up. For detailed information please see the description of the **Tabular Declaration Editor**.

Trace

Provides commands for working with the *Trace* editor. These commands are available in the Trace menu when the editor is active.

- Download Trace
- Start/Stop Trace
- Reset Trigger
- Cursor
- Mouse scrolling
- Mouse zooming
- Reset View
- Compress
- Stretch
- Multi-Channel
- Load Trace...
- Save Trace...

For further information, see **Trace Editor**.

8.Editors

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 d% 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**.

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.

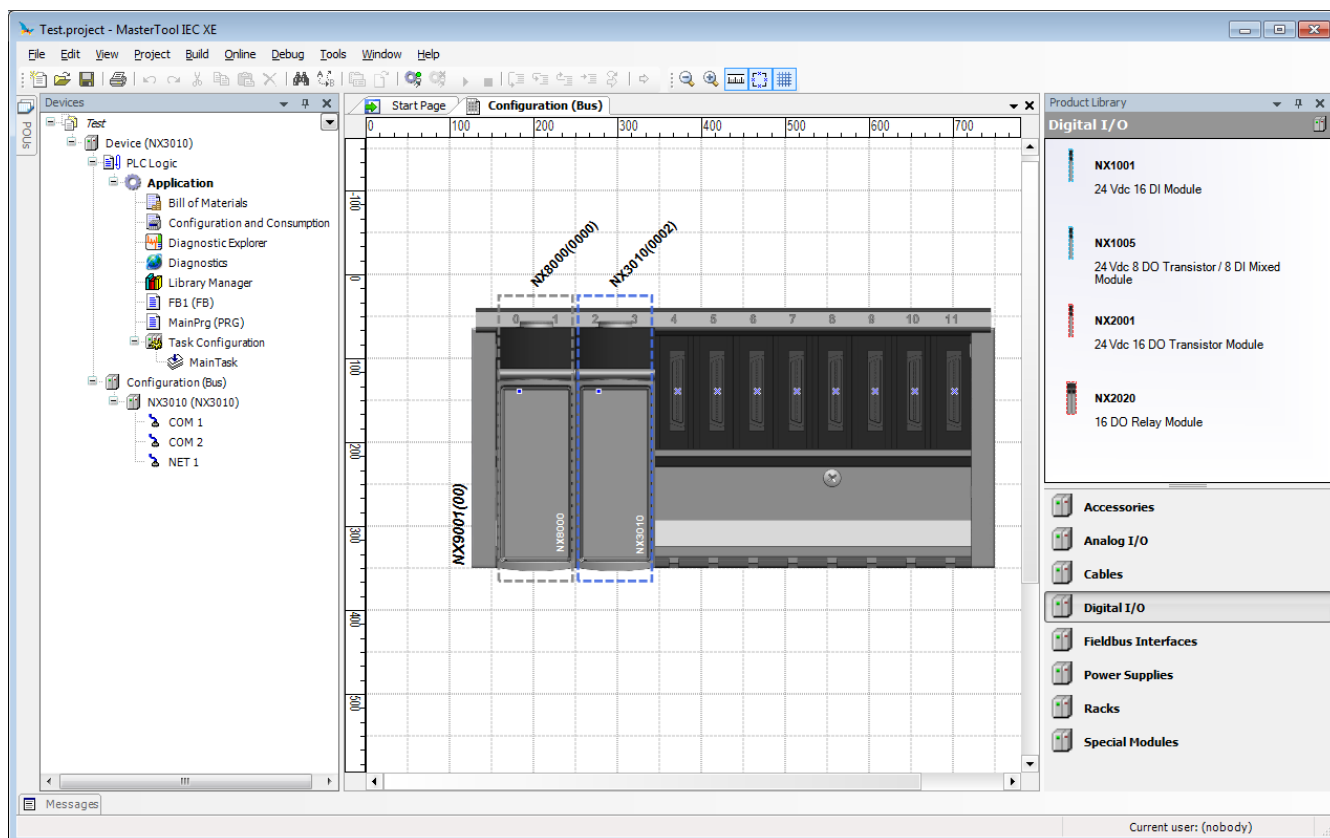


Figure 8-1. Bus Editor

Add Device

To add a device to the bus, it must be selected in the *Products Library* and dragged to the desired spot insertion.

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.

Digital I/O Module Editor

By adding a digital I/O module to the bus there are three possible configuration screens for it.

Process Data

Process data are the variables used by the I/O module for access and control, as shown in Figure 8-2:

Process Data

Module Parameters

Bus I/O Mapping

Select the Outputs

Name	Type
☐ HSC Input 00 Command	
High Speed Counter Input 00 Command	BYTE
☐ HSC Input 00 Preset Value	
High Speed Counter Input 00 Preset Value	DINT
☐ HSC Input 01 Command	
High Speed Counter Input 01 Command	BYTE
☐ HSC Input 01 Preset Value	
High Speed Counter Input 01 Preset Value	DINT
☐ Pulse-Catch Reset	
Pulse-Catch Reset - Byte 0	BYTE
Pulse-Catch Reset - Byte 1	BYTE

Select the Inputs

Name	Type
☒ Digital Inputs	
Digital Inputs - Byte 0	BYTE
Digital Inputs - Byte 1	BYTE
☐ HSC Input 00 Status	
High Speed Counter Input 00 Status	BYTE
☐ HSC Input 00 Current Value	
High Speed Counter Input 00 Current	DINT
☐ HSC Input 01 Status	
High Speed Counter Input 01 Status	BYTE
☐ HSC Input 01 Current Value	
High Speed Counter Input 01 Current	DINT
☐ Input 02 Period	
Input 02 Period	DWORD

Figure 8-2. Process Data

Information regarding each parameter is described in the current module Technical Characteristics.

Module Parameters

The modules parameters are configured through the screen shown in Figure 8-3:

Process Data Module Parameters Bus I/O Mapping		
Name	Value	Comment
Operating Mode	Mode 0	Set special features configuration mode.
Input Filter Enable Mask		Enable or disable Input Filter feature.
Input 00	FALSE	
Input 01	FALSE	
Input 02	FALSE	
Input 03	FALSE	
Input 04	FALSE	
Input 05	FALSE	
Input 06	FALSE	
Input 07	FALSE	
Input 10	FALSE	
Input 11	FALSE	
Input 12	FALSE	
Input 13	FALSE	
Input 14	FALSE	
Input 15	FALSE	
Input 16	FALSE	
Input 17	FALSE	
Input Filter Time Constant	7	Set Input Filter Time Constant (ms).
Pulse-Catch Enable Mask		Enable or disable Pulse-Catch feature.
Pulse-Catch Elongation Time	50	Set Pulse-Catch Elongation Time (ms).
Period Measurement Enable Mask		Enable or disable Period Measurement feature (available).
Input 02	FALSE	
%Q Start Address of Module Diagnostics Area	22040	Define starting address of Module Diagnostics Area.

Figure 8-3. Module Parameters

The information about each parameters is described in the current module Technical Characteristics.

Bus: I/O Mapping

The Figure 8-4, shows the map of all the module inputs/outputs.

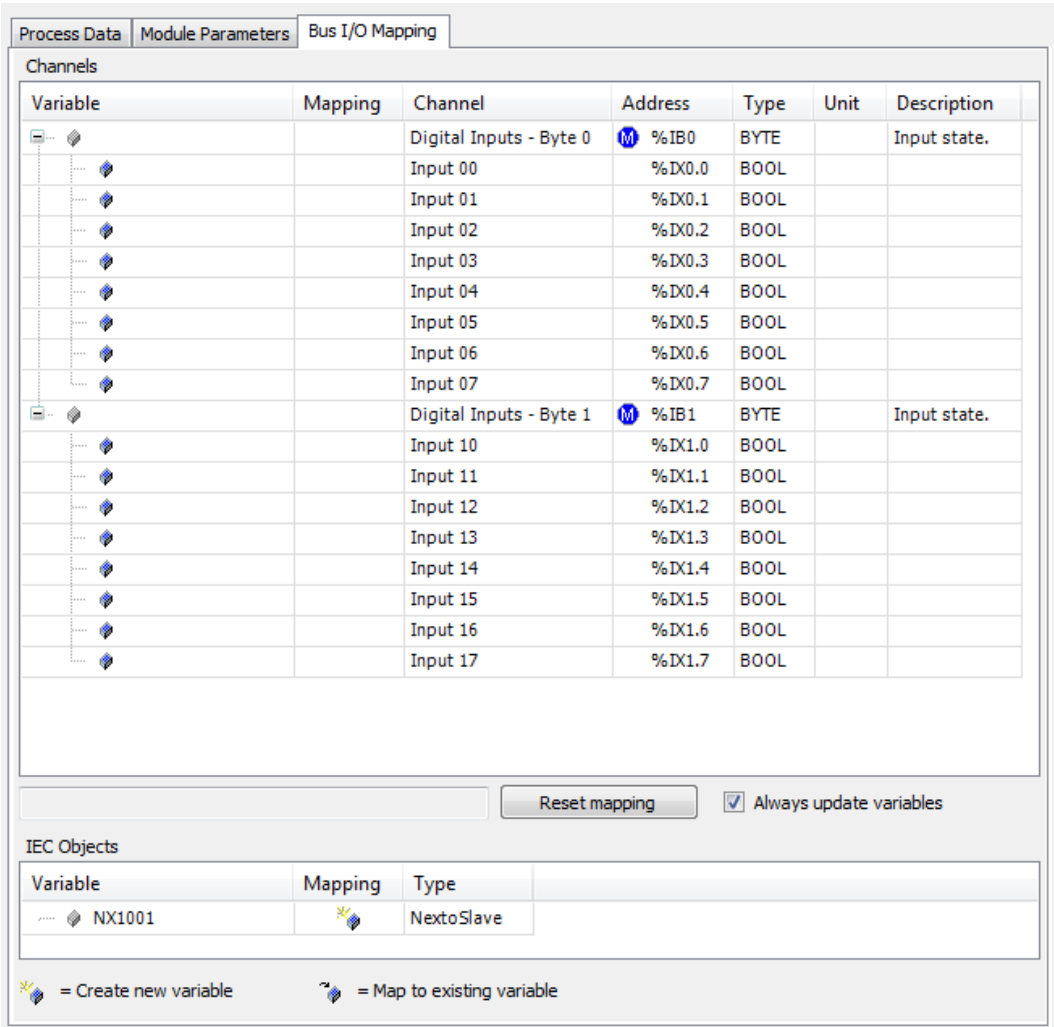


Figure 8-4. I/O Module Map

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 Figure 8-5. This list can also be used for purchase orders.

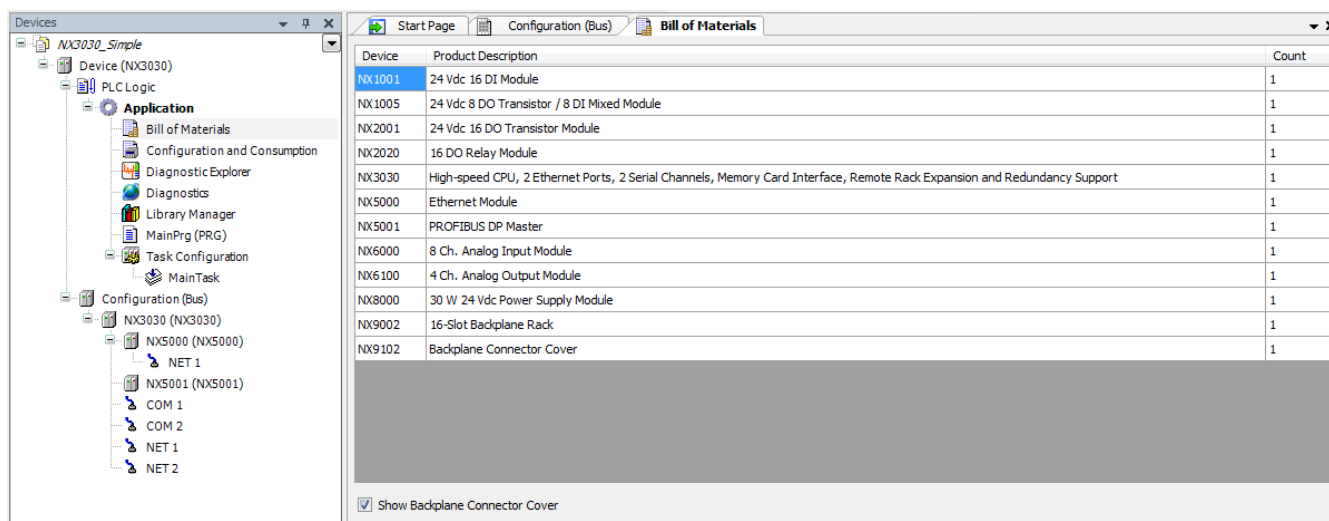


Figure 8-5. Bill of Materials

This list is available on the device tree for any project created from the *MasterTool 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.

Configuration and Consumption

Once configured the bus modules, MasterTool IEC XE allows the user to view the backplane complete configuration as shown in Figure.

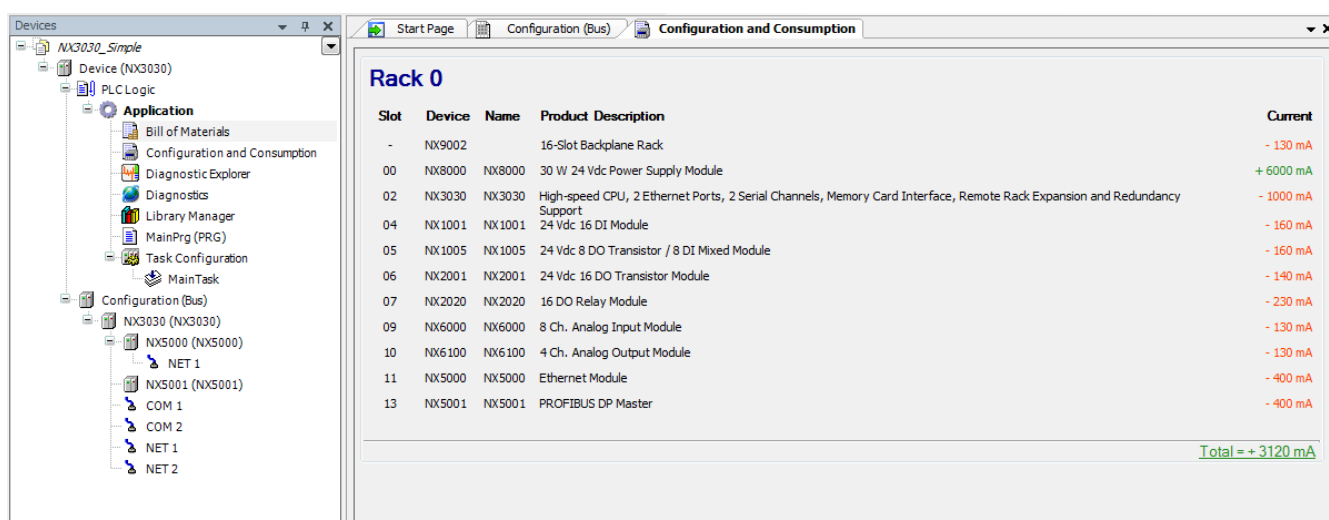


Figure 8-6. Configuration and Consumption

The configuration and bus consumption list is available in the device tree for any project created from the *MasterTool 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.

Programming Language Editors

For details see **Programming Language Editors (CFC, SFC, ST, FBD/LD/IL)**, in IEC 61131 Programming Manual.

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.

Textual Declaration Editor

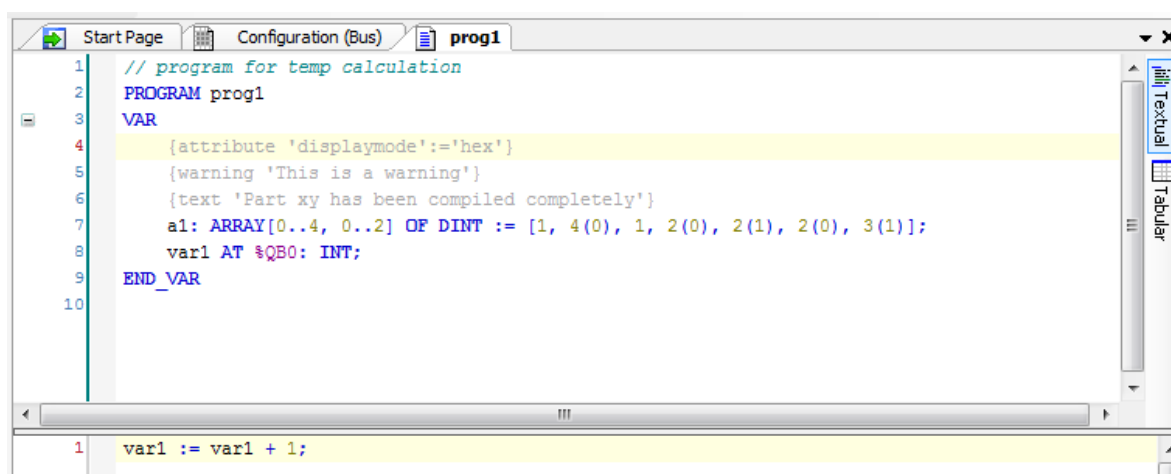


Figure 8-7. Textual Editor View

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.

Tabular Declaration Editor

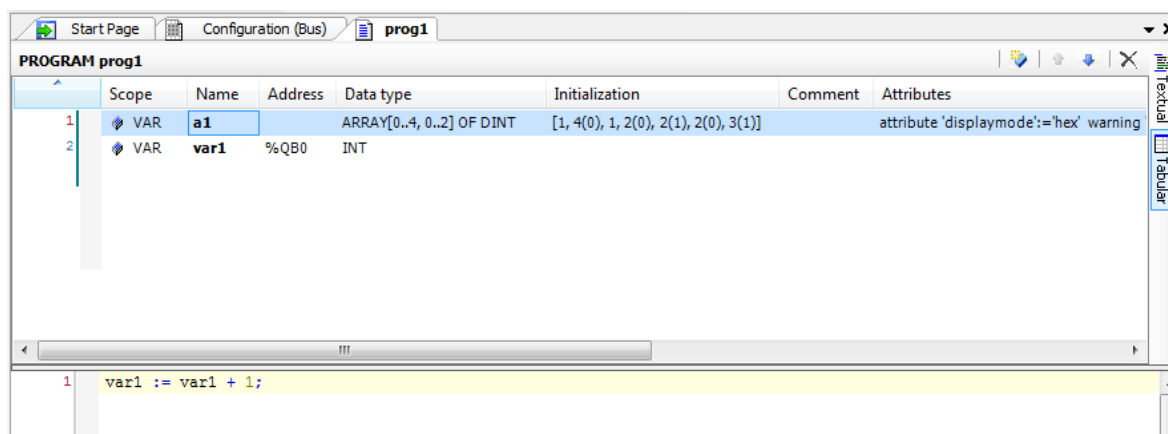


Figure 8-8. Tabular Editor View

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.

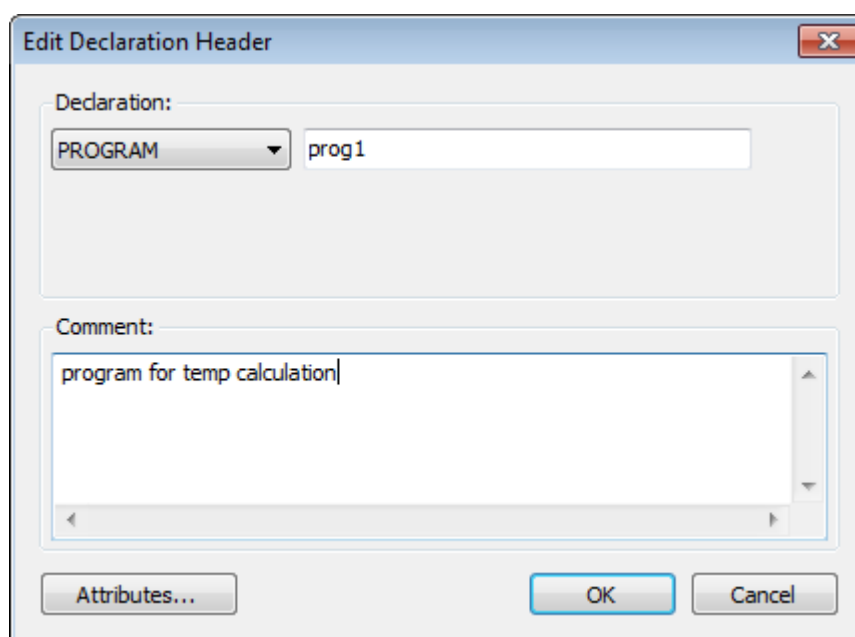


Figure 8-9. Edit Declaration Header Dialog

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.

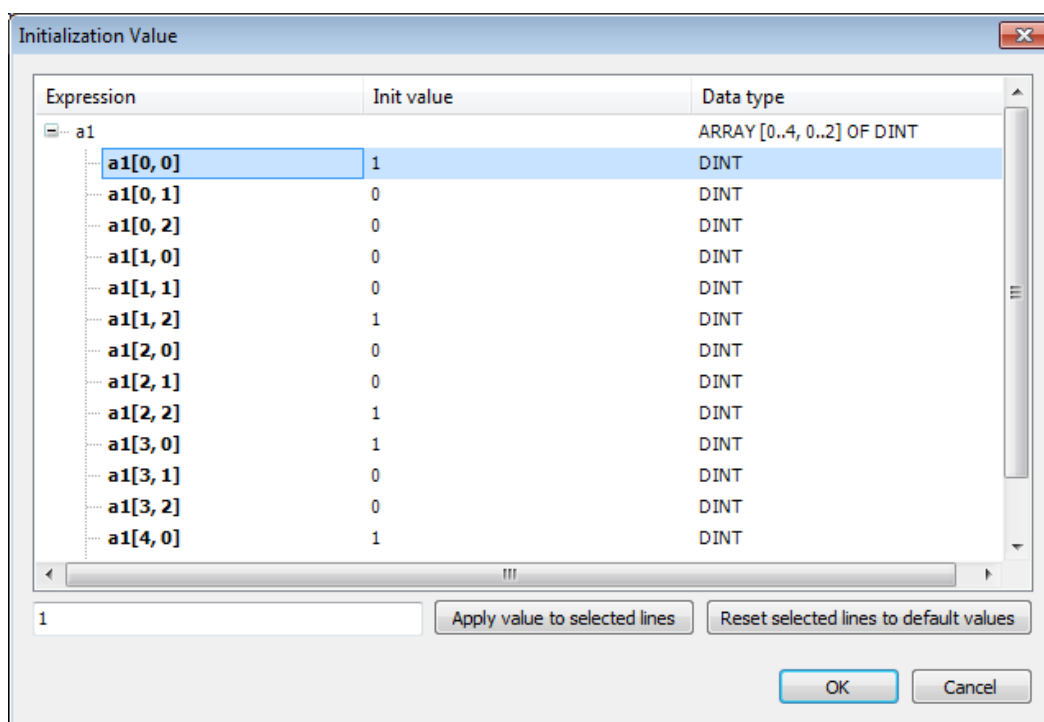


Figure 8-10. Initialization Value Dialog

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.

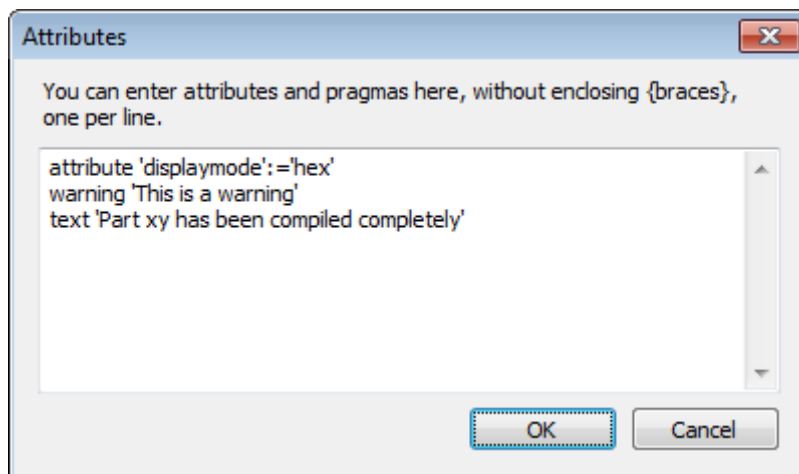


Figure 8-11. Attributes

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 ().

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 fbinst and fbinst2 Figure 8-12). 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 Figure 8-12.

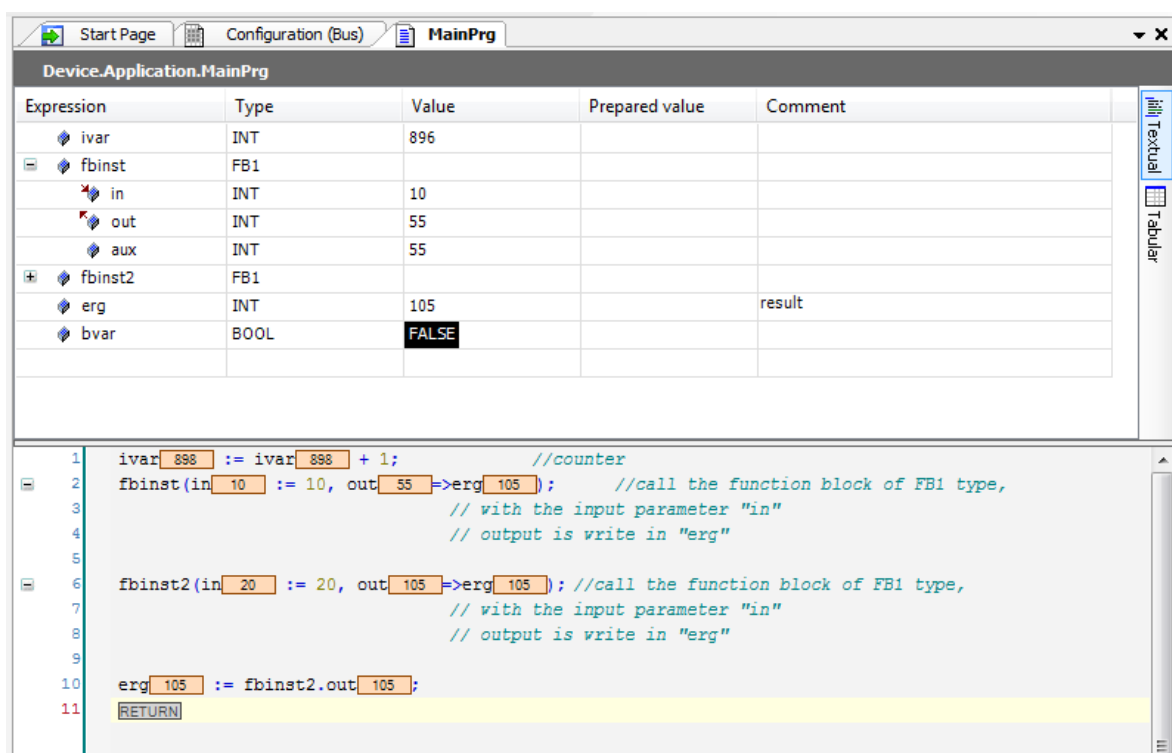


Figure 8-12. Declaration Editor in the Upper Part of a program object MainPrg, Online View

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.
- *Information*: General information on the device (name, provider, version etc.)

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.

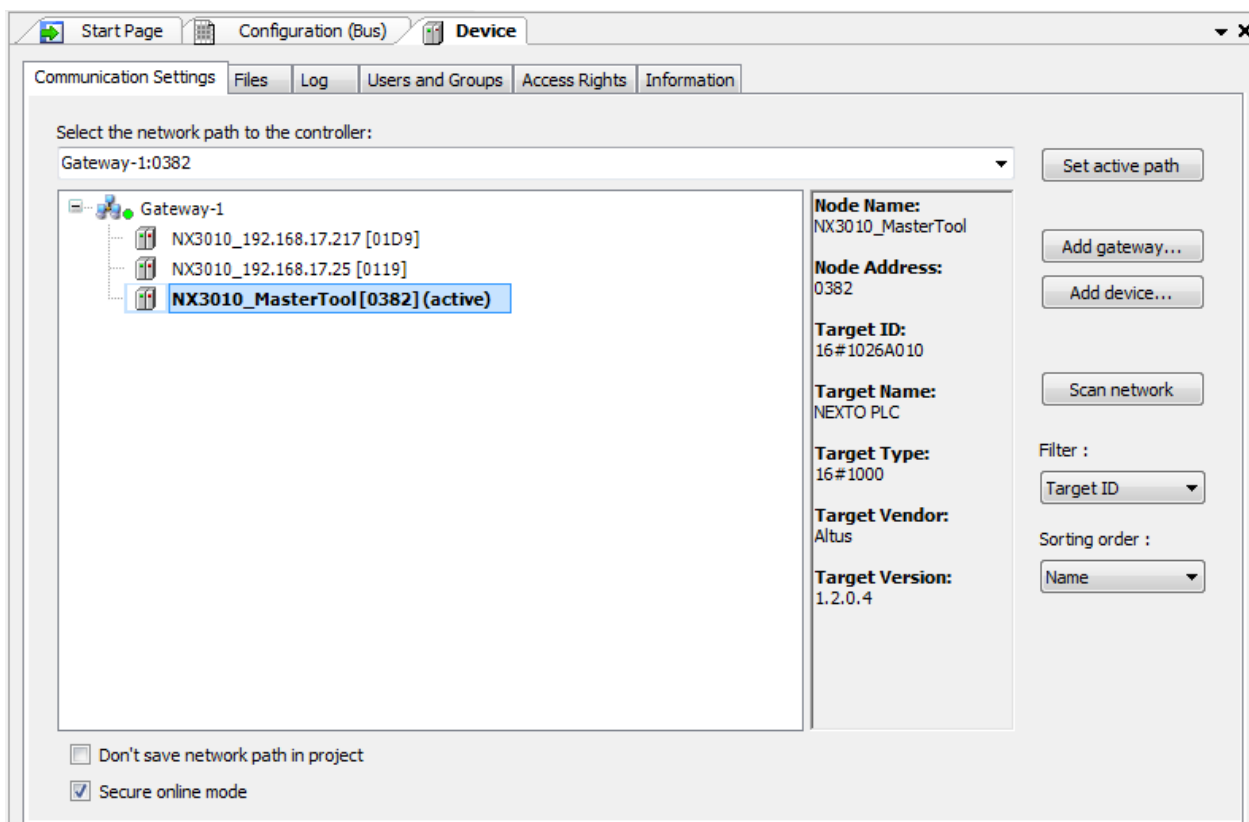


Figure 8-13. Device Dialog, Communication Settings

The dialog contains a bipartite window, which in the left part shows in a tree structure the currently configured gateway channels and in the right part the corresponding data and information.

When creating the first project at your local system, there will be no gateway configured. To add a local gateway click *Add Gateway...* and after that click *OK* (*Gateway* window).

By selecting a gateway, the user may see, in the right part of the window: the *Node Name*, *Node Address*, *Driver*, *IP-Address* and *Port*.

MasterTool IEC XE Gateway Service: The MasterTool IEC XE Gateway Server is started automatically at system start as a service and this service is represented by a control icon in the system bar. Icon ??? indicates a running, icon ??? indicates a 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.

As long as the gateway is properly running, a green bullet  is displayed before the entry, otherwise a red one . The bullet stays grey, if the gateway has not been contacted yet (depending on some communication protocols, it is not allowed to poll the gateway, so the status cannot be displayed).

Indented below the gateway entry (open/close via the +/- sign) you will see entries for all devices which are reachable through this gateway. In case there is no entry, use the button *Scan network* or *Add Device...*. Regard that those entries are stored locally on your system and not with the project.

The device entries are preceded by a  symbol. Entries with a target ID different to that of the device currently configured in the project, are displayed in grey font.

In the right window part of the device you see the respective *Node Name*, *Node Address*, *Target ID*, *Target Name*, *Target Type*, *Target Vendor* and *Target Version*.

The display of the gateway and devices nodes' tree can be controlled by filter and sorting functions. See the selection boxes in the right part of the dialog:

- *Filter*: The list can be reduced to all devices with a Target ID matching with that of the device currently configured in the project.
- *Sorting order*: The list can be sorted according to the Name resp. Node Address in alphabetical resp. ascending order.

Below the node tree, the following commands are available:

- *Do not save network path in project*: Activate this option, if the current network path definition should not be stored in the project, but in the local option settings on your computer. So the path setting will be restored if the project is reopened on the same computer, but will have to be redefined if the project is used on another system.
- *Secure online mode*: Activate this option if the user for security reasons should be prompted for confirmation when selecting one of the following online commands: *Force values*, *Release force list for < application >*, *Single Cycle*, *Start < application >*, *Stop < application >* and *Write Values*.

In the context menu of the node tree objects we can find the following commands:

- *Change Node Name*: This command opens the same-named dialog where is possible to view the current name and add a new one to the selected device.
- *Delete Selected Device*: This command removes the currently selected device or gateway.

In the right part of the dialog there are the buttons:

- *Set active path*: This command sets the communication channel, currently selected in the tree below the gateway, as the active path to the controller. This means that all actions concerning communication will exactly refer to this channel. The command will also be executed on a double-click on the channel entry. The currently active path will be displayed bold in the list, and “(active)” will be appended:  **NX3010_MasterTool[0382](active)**.

If a device has been set as the active path, but was not reachable during the last scan network, it will be drawn in italics.

NOTE: The information on which device was recently set as “active path” will be stored on the local system and can be automatically restored at a *Login* with not yet configured communication settings.

- *Add 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.

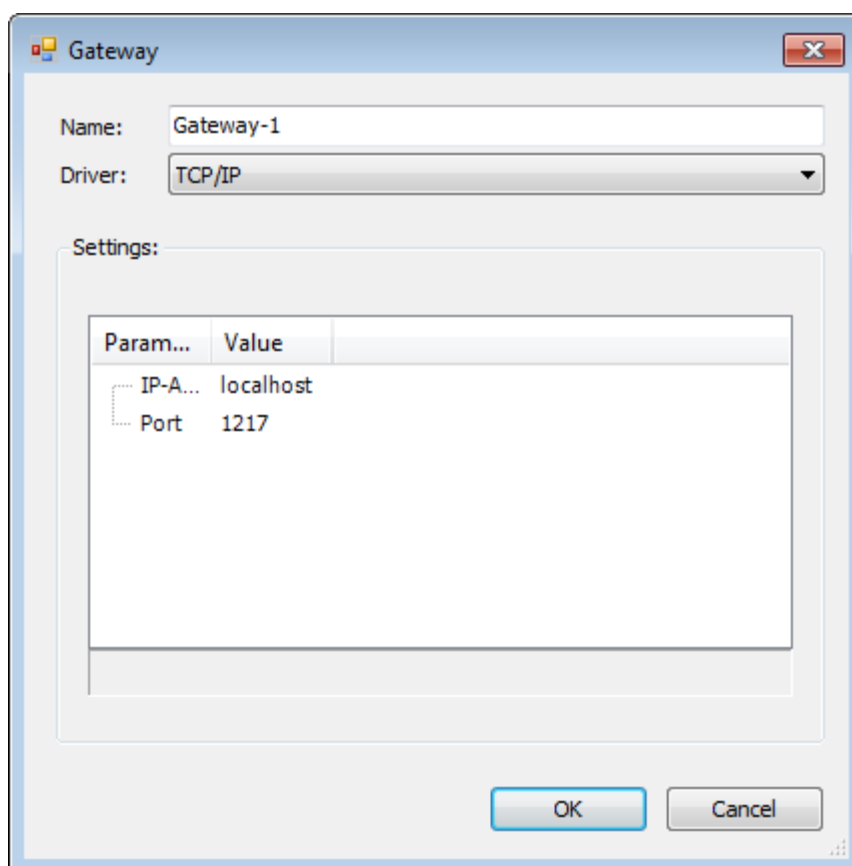


Figure 8-14. Gateway Dialog

ATTENTION:

The proper configuration of the gateway requires detailed knowledge. In case of doubt leave unchanged the default configuration settings.

- *Add device*: This command opens a dialog titled *Add device*, where you can manually define a device to be added to the currently selected gateway entry.

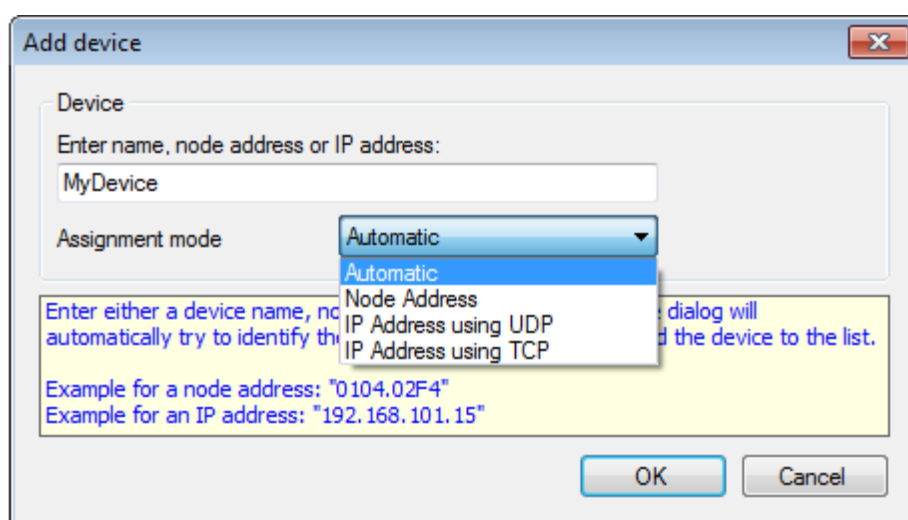


Figure 8-15. Add Device Dialog

You can specify the device to be added by its name (example: MyDevice), node address (example: "0104.02F4") or IP address (example "192.168.101.15").

Depending on whether you specify a name or an address, you have to set the appropriate Assignment mode that is how the system should process to add the desired device. In most cases your specification will be unambiguous, so that the "automatic" mode will work. The other two modes are only needed, when you have to precise the type of your entry, because it is not evidently a name or an address. Example: You want to add a device with hex node address "AFFE". "AFFE" also could be a target name, so, to make the specification clear, set option *Node Address*.

- *Automatic*: The dialog will automatically search the correct mechanism to find and add the device to the list.
- *Node Address*: If this option is selected, a unique node address (example: 02F4) must be specified. The network then will be checked for the availability of a device with that address and will add it to the list.
- *IP Address using UDP*: If this option is selected, a unique IP address must be specified with the UDP port of the attached device (eg 192.168.192.15:1749). The network then will be scanned for the respective device in order to get its node address and to add it to the list.
- *IP Address using TCP*: If this option is selected, a unique IP address (eg 192.168.101.15) must be specified. The network then will be scanned for the respective device in order to get its node address and to add it to the list.

If you specify a device, which is not available in the network, the device entry anyway will be added to the list with a default name ("NewDevice") and address ("FFFF.FFFF.FFFF"). You can use the *Resolve Address* command to identify the device when it is connected later on. Please note that any network scan will delete this device when it is not available at the time of the scan.

- *Scan network*: This command starts a search for available devices in your local network and the configuration tree will be updated accordingly. The command will also be executed on a double-click on the gateway entry. The found devices will be listed below that gateway entry, in the tree of which an entry has currently been selected when calling the command.

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 8-16 where Gateway (GW0) of the computer (PC10) is being used to access the device from same subnet (PLC20). In Figure 8-16 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.

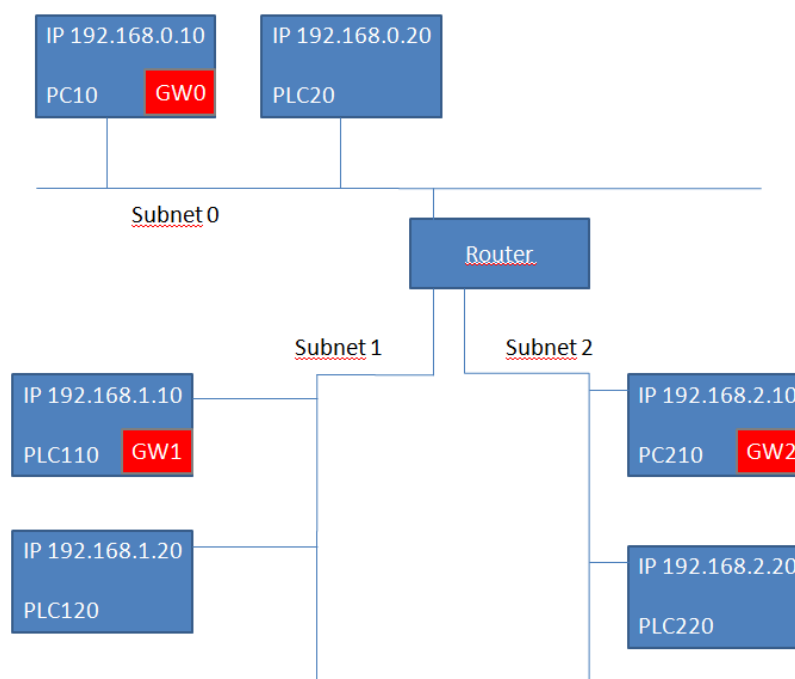


Figure 8-16. Remote Gateway

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 CP 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 following picture, Figure 8-16, 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 of your. 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.

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.

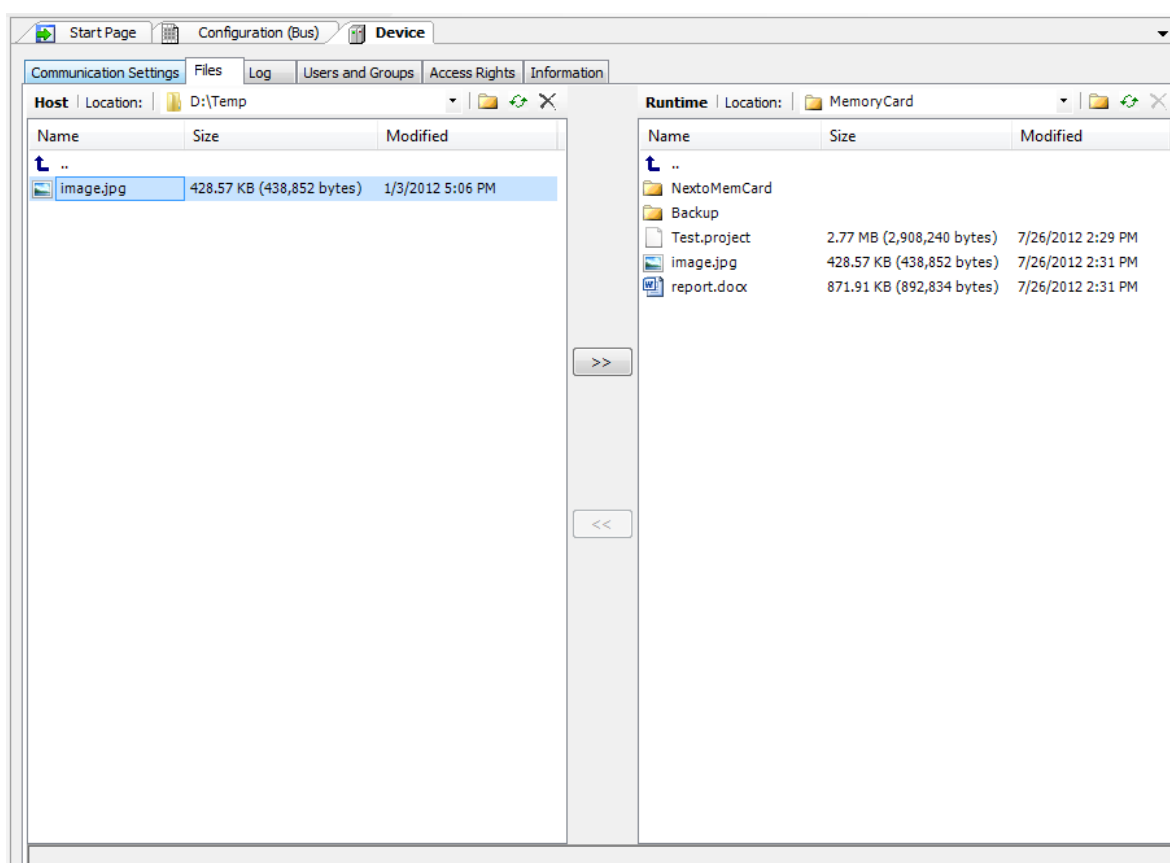


Figure 8-17. Device dialog, Files

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.

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.

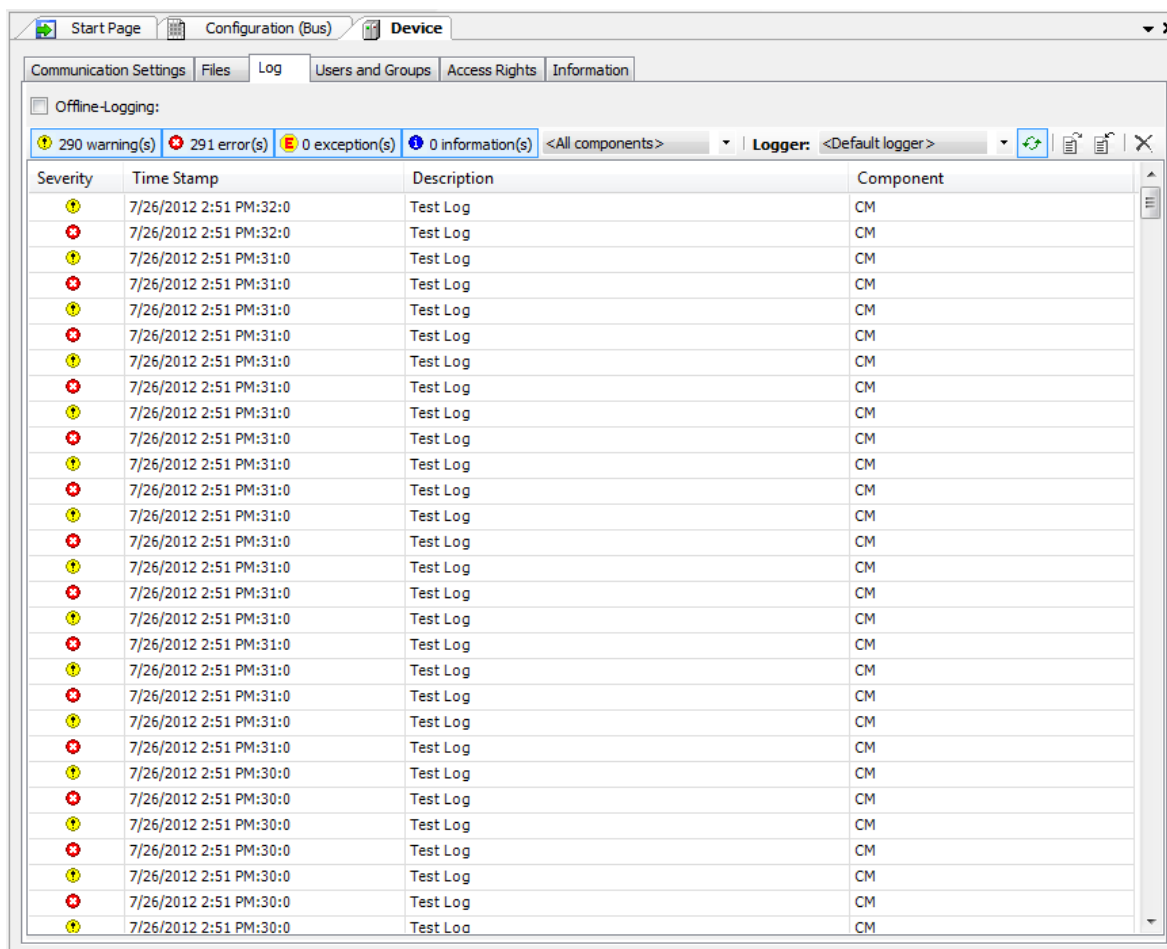


Figure 8-18. Device Dialog, Log

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.



- **Time Stamp:** Date and Time: MM/DD/YY h:mm AM/PM:ss:ms. Example: "2/15/2012 7:05 AM:20:0".
- **Description:** description of the event, for example: "Import function failed of Component <CmpFileTransfer>".
- **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 loggings. 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 manually can update the list by using  button.

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.

DUT Editor

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 POU's view.

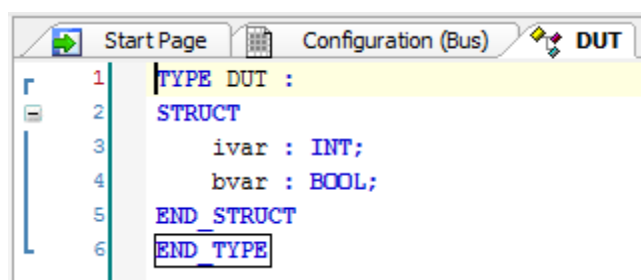


Figure 8-19. DUT Editor Window

FBD/LD/IL Editor

A combined editor is available for editing POU's in the languages FBD (Function Block Diagram), LD (Ladder Diagram) and IL (Instruction List).

This means a common set of commands and elements is used and an automatic internal conversion between the three languages is done. In offline mode the programmer always can switch to one of the other editor views.

Notice anyway, that there are some special elements, which cannot get converted and thus will only be displayed in the appropriate language. For further details see **FBD/LD/IL Editor Commands**. In addition, there are some constructs, which cannot get converted unambiguously between IL and FBD and therefore will be “normalized” at a conversion back to FBD. This concerns: Negation of expressions and explicit/implicit output assignments.

The editor will open in a bipartite window, when editing an object programmed in FBD/LD/IL, the upper part containing a declaration editor.

The programming language for a new object is specified when creating the object via command *Add Object*.

For further information see the IEC 61131 Programming Manual.

Global Variables List Editor

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.

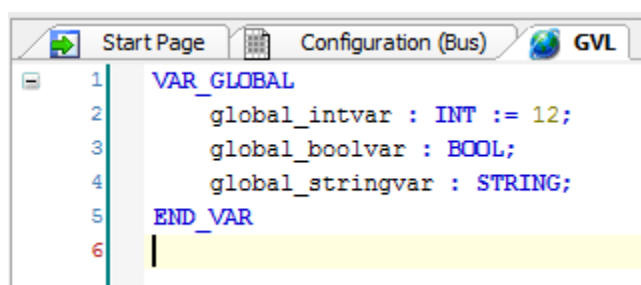


Figure 8-20. GVL Editor

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.

NOTE: The *Library Repository* dialog is only available if predefined feature sets chosen by the user are *Professional* or the option *Enable repository dialog* is enabled. For further information about features, see **Features**.

The object is available in the devices tree to any project and is created from the *MasterTool 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.

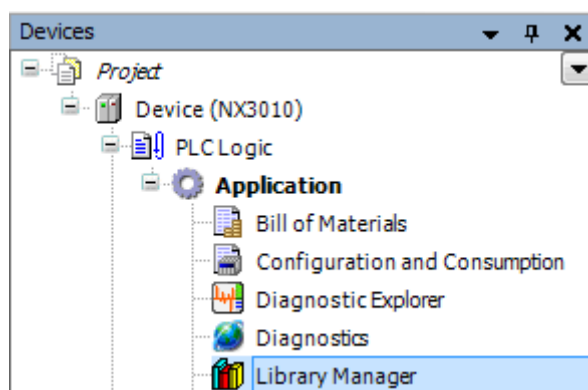


Figure 8-21. Library Manager Object in the POU's View Window

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.

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.

NOTE: Some dialogs and *Library Manager* editor window functionalities is only available if predefined feature sets chosen by the user are *Professional* or if certain options are enabled. For further information about features, see **Features**.

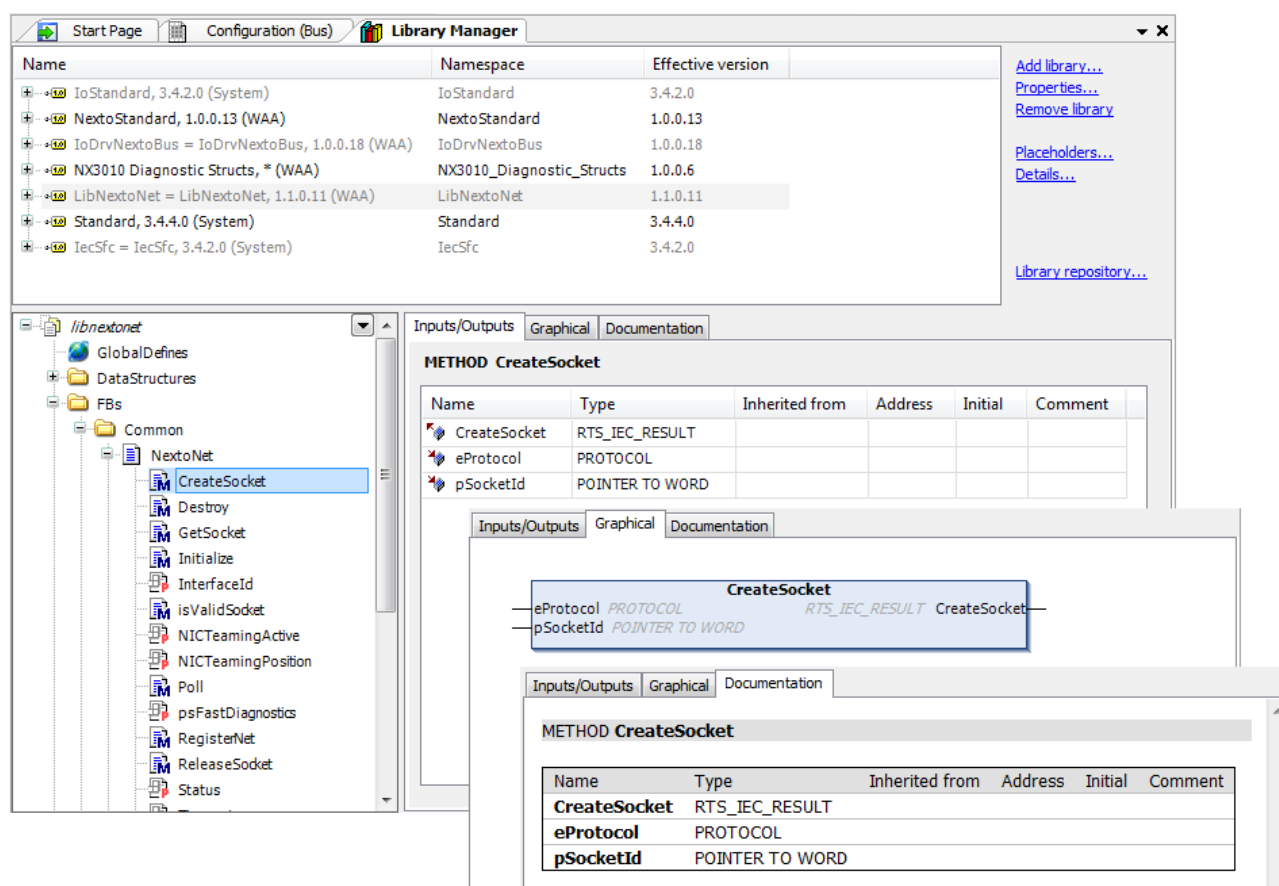


Figure 8-22. Library Manager Editor

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 plurally available 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.

Library included in the project.
Tab "Documentation".

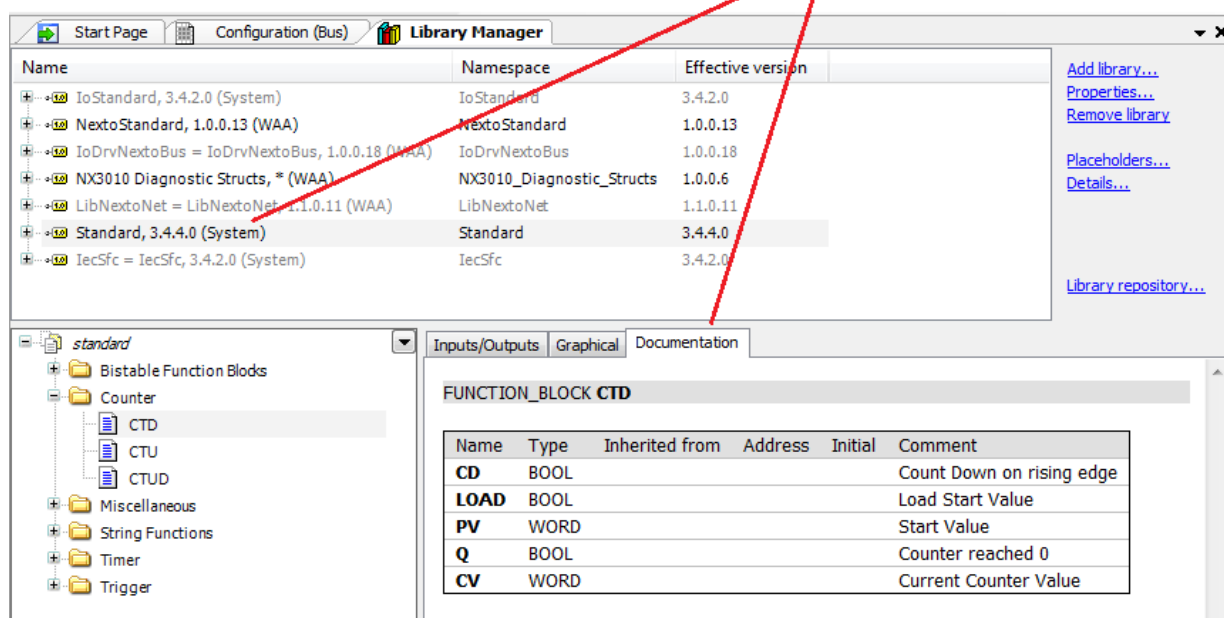


Figure 8-23. Commented Library Modules

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.
- *Properties*: for settings on the namespace and - with notice that the library will later be available as a "referenced library" in another project - for settings on version handling, visibility and accessing.
- *Remove library*: the library currently selected in the libraries list will be removed from the project.
- *Library Repository*: for defining library locations and for installing or uninstalling libraries. The user will be notified if trying to insert a library, which is already available in the project.

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.

Task Editor

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 *MasterTool 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.

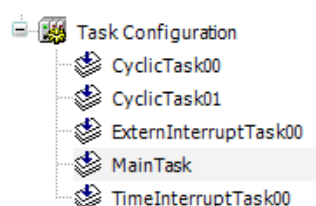


Figure 8-24. Task Configuration in Devices Tree Below an Application

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 POU's 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.

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.

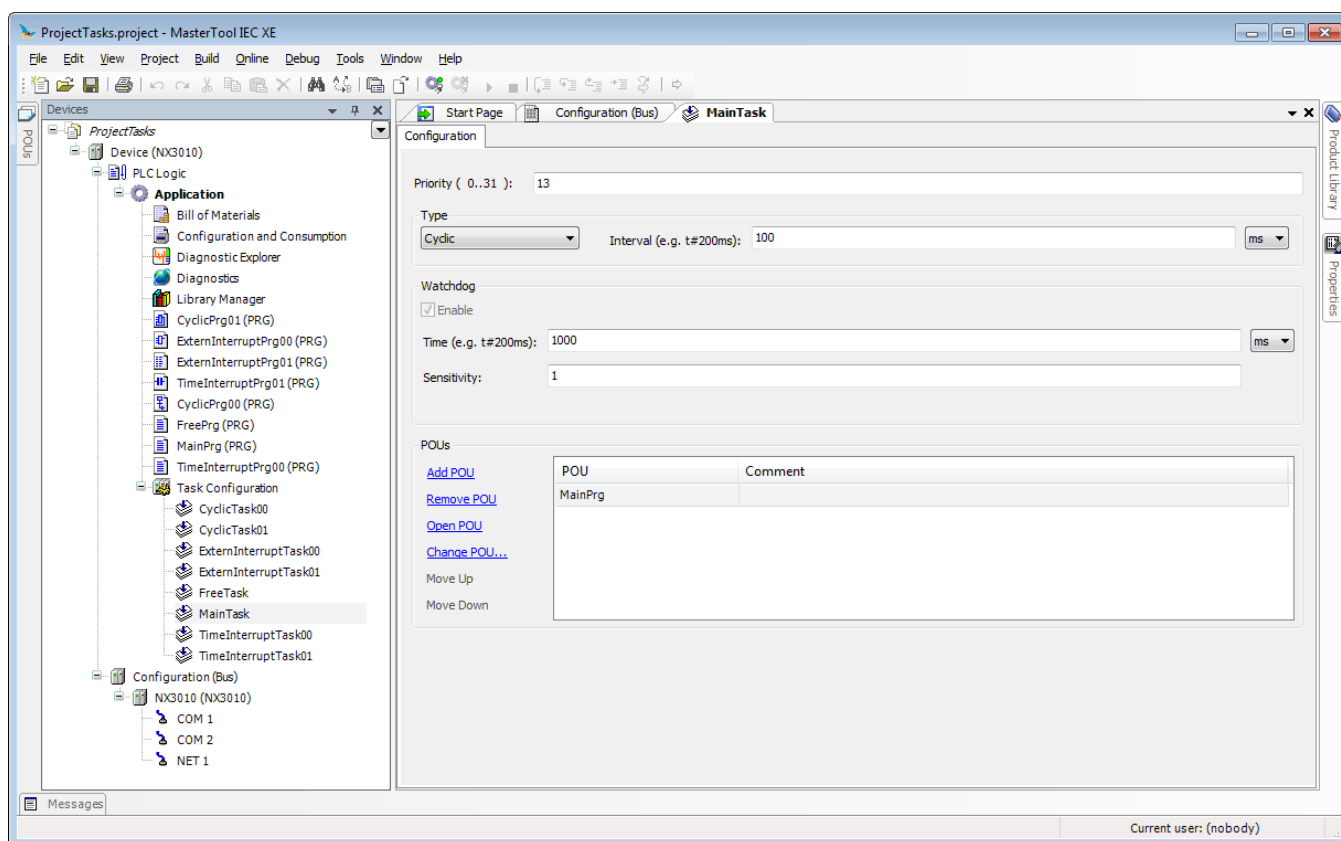


Figure 8-25. Task Editor, Dialog for Configuration of a Task

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).
- *Events*: System events dialog for linking POU calls to system events.

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.

Properties Dialog

When the top entry in the task configuration tree is selected, the *Properties* dialog will be opened in the task editor window. Figure 8-26 shows the *Properties* window of a CPU NX3010.

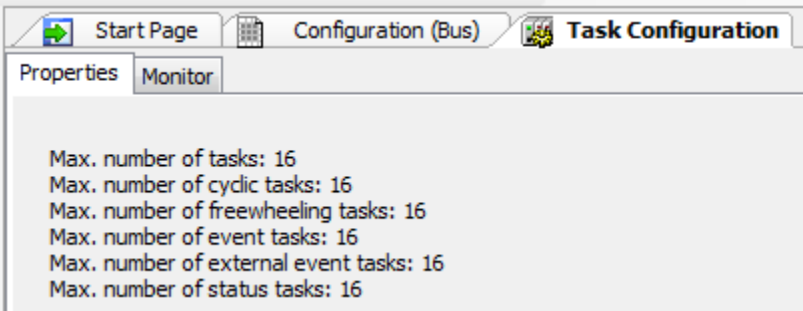


Figure 8-26. Task Configuration Properties Dialog

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.

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.

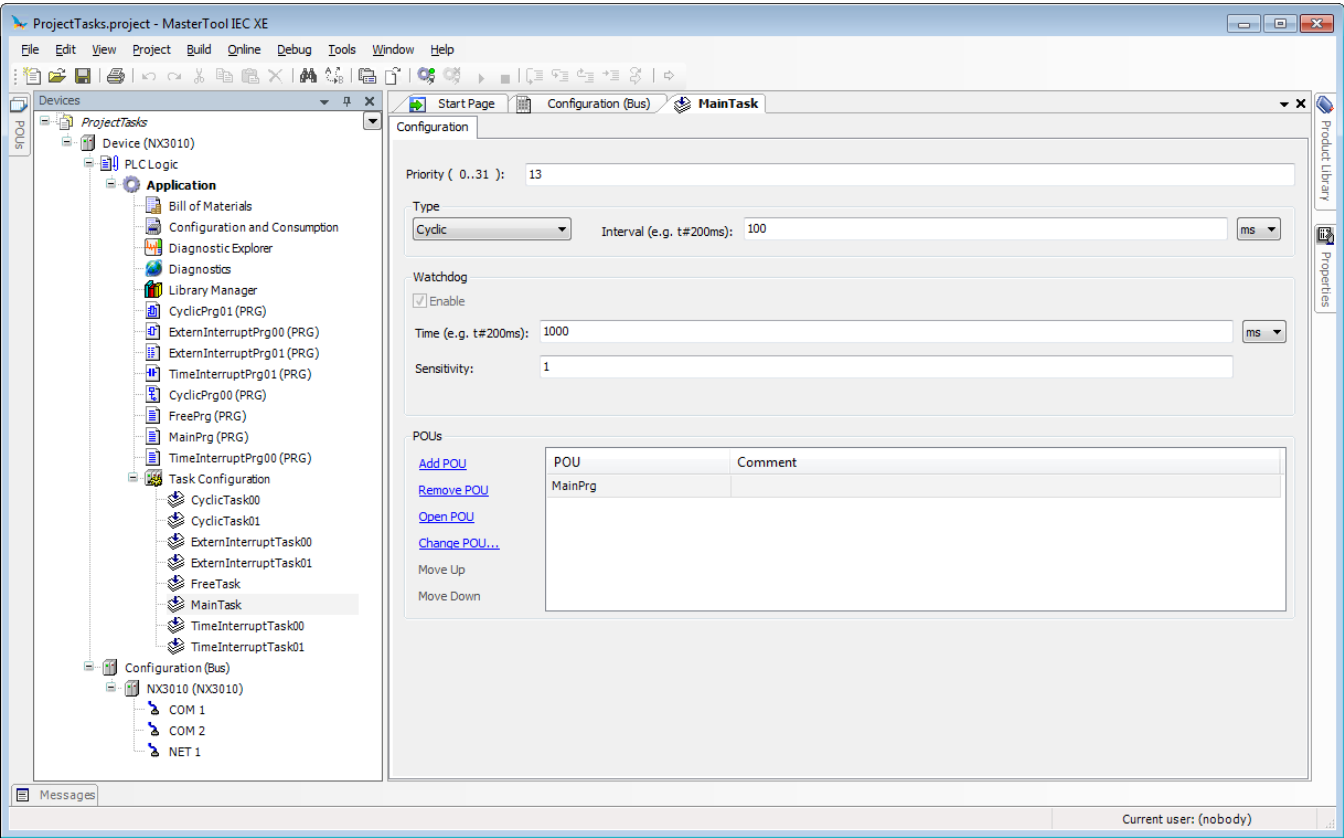


Figure 8-27. Configuration Dialog for a Task

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.

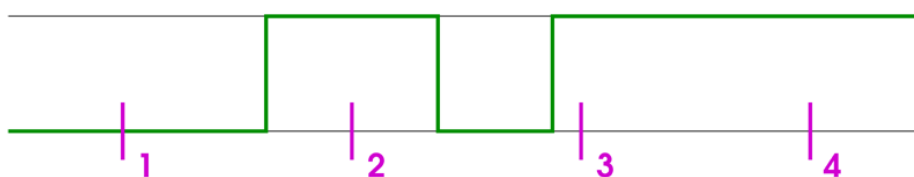


Figure 8-28. Resulting Behavior of the Task in Reaction to an Event (Green Line)

At sampling points 1-4 (magenta) tasks of different types show different reaction: (Figure 8-28).

Behavior at point	1	2	3	4
Status	no start	Start	Start	Start
Event	no start	start	no start	no start

Table 8-1. 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 [μs] 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 8-2. 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 POU, 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.

Task Editor in Online Mode

Which task is being Processed?

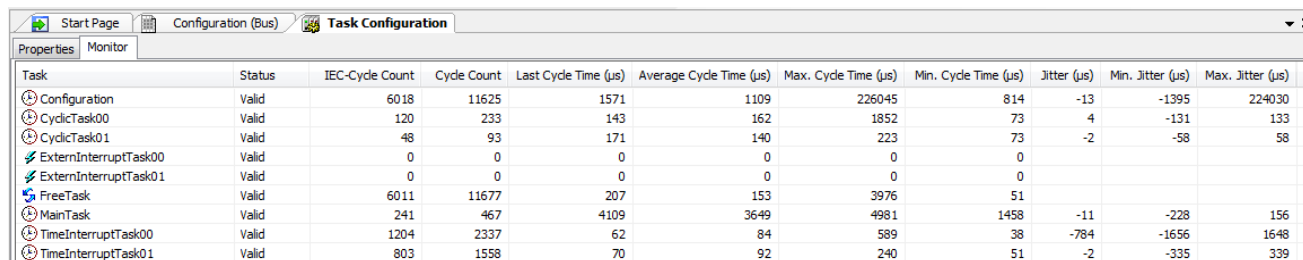
For the execution of the tasks defined in the Task Configuration the following rules apply:

- 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 POU's window, that one directly assigned to the application will be executed.

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.



Task	Status	IEC-Cycle Count	Cycle Count	Last Cycle Time (µs)	Average Cycle Time (µs)	Max. Cycle Time (µs)	Min. Cycle Time (µs)	Jitter (µs)	Min. Jitter (µs)	Max. Jitter (µs)
Configuration	Valid	6018	11625	1571	1109	226045	814	-13	-1395	224030
CyclicTask00	Valid	120	233	143	162	1852	73	4	-131	133
CyclicTask01	Valid	48	93	171	140	223	73	-2	-58	58
ExternInterruptTask00	Valid	0	0	0	0	0	0			
ExternInterruptTask01	Valid	0	0	0	0	0	0			
FreeTask	Valid	6011	11677	207	153	3976	51			
MainTask	Valid	241	467	4109	3649	4981	1458	-11	-228	156
TimeInterruptTask00	Valid	1204	2337	62	84	589	38	-784	-1656	1648
TimeInterruptTask01	Valid	803	1558	70	92	240	51	-2	-335	339

Figure 8-29. Task Configuration, Monitoring

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 having started the application. "0" if the function is not supported by the target system.
Cycle Count	Number of already run cycles. Depending on the target system this can be equal to the IEC-Cycle Count if cycles are even counted when the application is not running.
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 (µs)	Minimum measured runtime of all cycles in µs.
Jitter (µs)	Last measured jitter* in µs.
Min. Jitter (µs)	Minimum measured jitter* in µs.
Max. Jitter (µs)	Maximum measured jitter* in µs.

Table 8-3. Task Information

* 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.

Task Configuration Commands

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.

Trace Editor

Trace

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.

The configuration as well as the display of the trace sampling is done in the Trace Editor dialogs and views. Multiple variables can be traced and displayed at the same time, if desired in different views.

In order to start trace sampling according to the current trace configuration in online mode, you have to download the configuration to the runtime system. The graphs of the trace variables will be displayed in the *Trace Editor* window and you might store them to an external file, which later can be reloaded to the editor.

Commands for accessing the configuration dialogs as well as commands for modifying the currently displayed section of the trace curves resp. graphs are available in the Editor window.

The readout of a trace may also be integrated within a visualization by use of the special visual element trace.

Available commands:

- Download Trace
- Start/Stop Trace
- Reset Trigger
- Cursor
- Mouse Scrolling
- Mouse Zooming
- Reset Visualization
- Compress
- Stretch
- Multi-Channel

Download Trace

This command will download the trace configuration in order to get the tracing activated on the device runtime system. This is necessary for the first tracing on an application and later always after any changes in the trace configuration or the application program.

Notice that an active Trace is indicated by a green background color of the Trace object in the *Devices* tree.

Start/Stop Trace

Symbols: 

This command toggles start and stop tracing.

If tracing currently is stopped, the  symbol is shown and performing the command means to restart the tracing, that is start to display the variables values as from the actual time.

If tracing currently is running, the  symbol is shown and performing the command means to stop the tracing. This means the display will freeze the currently viewed section of the graph(s).

Reset Trigger

Symbol: 

This command will reset the trace display after a trigger event has occurred or the trace has been stopped. The tracing (display) will continue with the actual values.

Cursor

Symbol: 

As long as this option is activated (Default), the actual position of the mouse-cursor in a coordinate system of the trace window will be always indicated in the status bar. You can move the cursor by clicking in the black triangle of the cursor and scroll it with pressed left mouse key. The same can be done by using left/right arrow key. All keyboard shortcuts listed in **Shortcuts for Trace**. The option will automatically be deactivated if one of the options *Mouse Scrolling* or *Mouse Zooming* is activated.

Mouse Scrolling

Symbol: 

This command activates the mouse-scrolling mode. If this mode is activated, the mouse-cursor will be displayed for a short moment as a double-arrow and you can scroll the time axis of the currently displayed trace by moving the mouse on the trace window. The same can be done with the left/right arrow key while pressing the ALT key. For vertical scrolling use the up and down arrow key. With additionally pressed CTRL key the vertical and horizontal scrolling is done in greater distances.

If the *Multi-Channel* option is enabled, you can scroll the time axis of all diagrams simultaneously with mouse or keypad (left/right arrow key while pressing the <ALT> key). For vertical scrolling of the selected diagram use the up and down arrow key. With additionally pressed <CTRL> key the vertical and horizontal scrolling is done in greater distances.

If you scroll out of the view range currently available in the trace buffer, “out of data” will be displayed in the trace window. To de-activate the mouse-scrolling mode, activate the Cursor mode.

To get back to the default appearance use command *Default Appearance*.

To zoom the display use command *Mouse Zooming*.

All keyboard shortcuts listed in **Shortcuts for Trace**.

Mouse Zooming

Symbol: 

This command activates the mouse-zooming mode. In this mode the mouse-cursor will get displayed as  and you can draw a rectangle in the trace window in order to re-define the area of the trace curves to be displayed. This area will then be viewed in a way that it fills the trace window.

To de-activate the mouse Zoom mode, activate the *Cursor* mode.

To get back to the default appearance use command *Default Appearance*.

Notice also the possibility to zoom the display of the trace window by using a “scroll mouse” or by using a keyboard.

- Scrolling the mouse wheel zooms the coordinate system along the x- and y-axes. The same can be done with the numeric keypad key + and key -.
- Scrolling the mouse wheel while pressing the <SHIFT> key only zooms along the x-axis. The same can be done with the numpad key + and key - while pressing the <SHIFT> key.
- Scrolling the mouse wheel while pressing the <CTRL> key only zooms along the y-axis. The same can be done with the numpad key + and key - while pressing the <CTRL> key.

All keyboard shortcuts listed in **Shortcuts for Trace**.

Reset Visualization

Symbol: 

This command restores the default settings of the graphics appearance after the it have been changed (by a zoom action, for example). The default settings are defined in the *Configuration* dialog.

Compress

Symbol: 

With this command the values shown for the Sampling Trace are compressed; i.e., after this command you can view the progression of the trace variables within a larger time frame. A multiple execution of the command is possible.

This command is the counterpart to *Stretch*.

Stretch

Symbol: 

With this command you can stretch (zoom) the values of the Sampling Trace that are shown.

With repeated stretches that follow one-after-another, the trace section displayed in the window will increasingly shrink in size.

This command is the counterpart to *Compress*.

Multi-Channel

With this command it is able to change the view of the trace diagram. Default view is a diagram with one x- and y-axis and all variables are visualized there. In multi-channel view every variable is visualized in its own diagram but with identical x-Axis. *Zooming* and *Scrolling* commands are affecting the x-axis of all diagrams simultaneously.

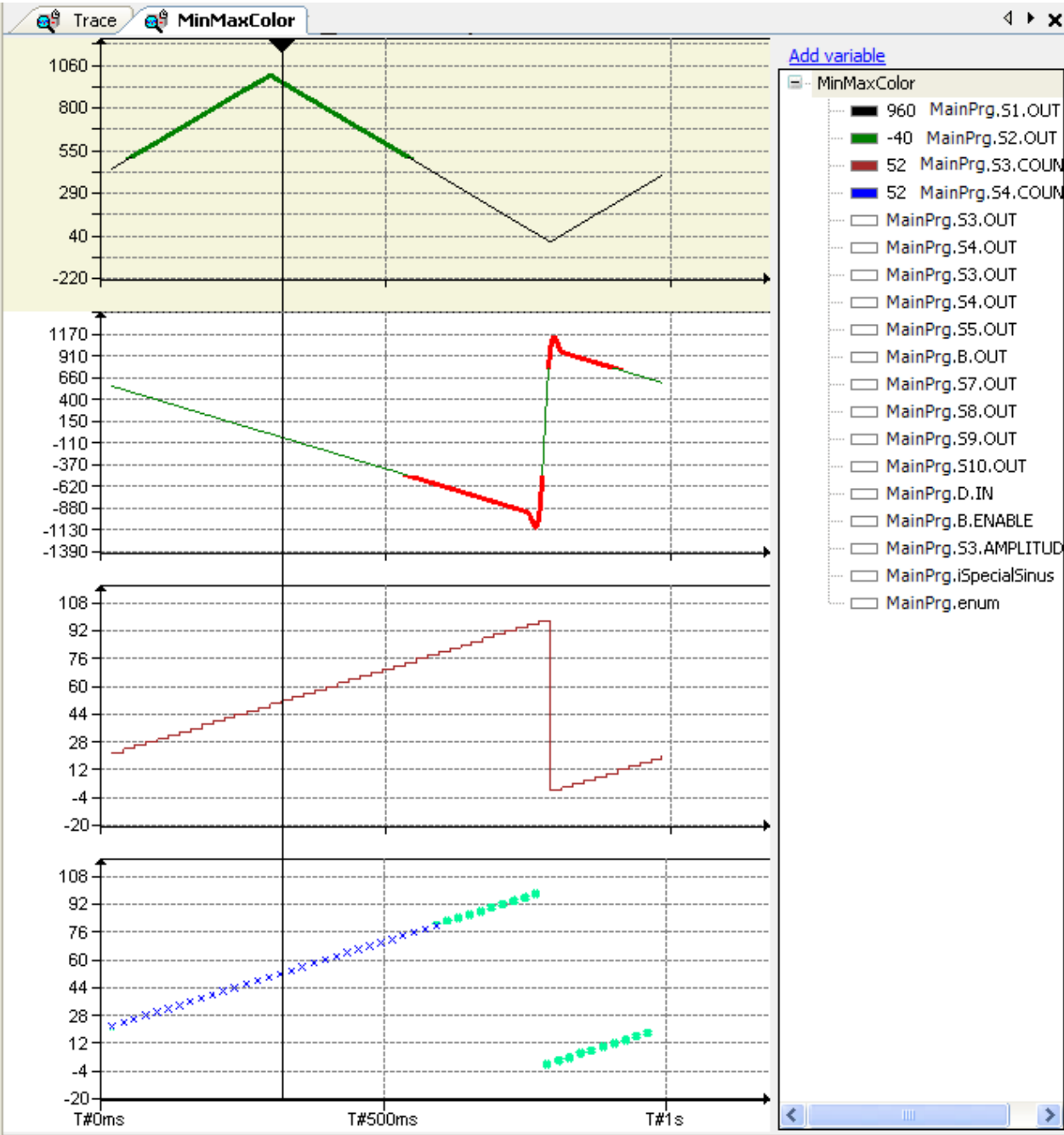


Figure 8-30. Multi-Channel

Shortcuts for Trace

NOTE: The graphs of the Trace diagram are changed in greater distances while pressing the <CTRL> key additionally to the following keys.

The trace editor and the commands in the trace menu provides following function keys in default mode:

Keyboard shortcuts	Trace diagram action
Left/right arrow key + <ALT>	Scroll the time axis of the trace horizontal
Up and down arrow key (without pressing <ALT>)	Scroll the time axis of the trace vertical

Keys + and -	Zooms the coordinate system along the x- and y-axes
Keys + and - + <SHIFT>	Zooms along the x-axis (horizontal)
Keys + and - + <CTRL>	Zooms along the y-axis (vertical)
Left/right arrow key	Scroll the cursor left and right

Table 8-4. Function Keys in Default Mode

With activated multi-channel option the using of the function keys has following effect:

Keyboard shortcuts	Keyboard shortcuts
Left/right arrow key + <ALT>	Scroll the time (horizontal) axis of all diagrams simultaneously
Up and down arrow key (with or without pressing the <ALT> key)	Scroll only the selected diagram vertical
Keys + and - (numeric keypad or keyboard)	Zooms the coordinate system along the x- and y-axes of all diagrams simultaneously
Keys + and - + <SHIFT> (numeric keypad or keyboard)	Zooms all diagrams along the x-axis (horizontal)
Keys + and - + <CTRL> (numeric keypad or keyboard)	Zooms all diagrams along the y-axis (vertical)
Left/right arrow key	Scroll the cursor left and right

Table 8-5. Function Keys in Multi-Channel Mode

Trace Editor

- Overview
- Commands and zoom functionality
- Configuration of the trace
- Configuration of variables
- Trace editor in online mode

Overview

The Trace Editor, is used to configure and to display “Traces”. A trace in this context is a sampling of the values of variables that is the online progression of the values read from the PLC over a certain time. Additionally a trigger can be set. If that had be done only the triggered values are recorded and displayed. For this purpose, the values are sampled in a MasterTool IEC XE buffer and can be visualized in the Trace Editor window in the form of a graph in a coordinate system of defined appearance.

Variables triggered simultaneously can be recorded in one Trace object. There the variables can be displayed in a variable specific coordinate system and possibly in multi-channel-mode. Traces of variables with different trigger have to be recorded in their own trace object. You can create any number of trace objects.

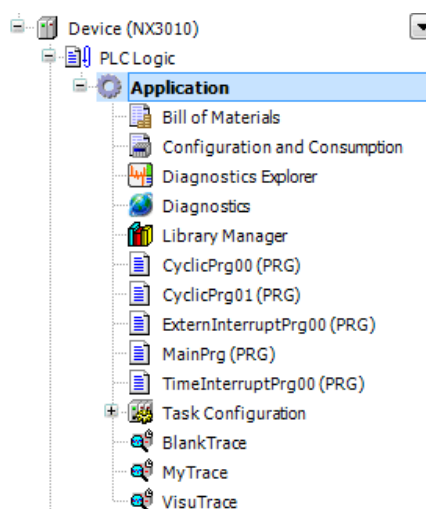


Figure 8-31. Project with Traces

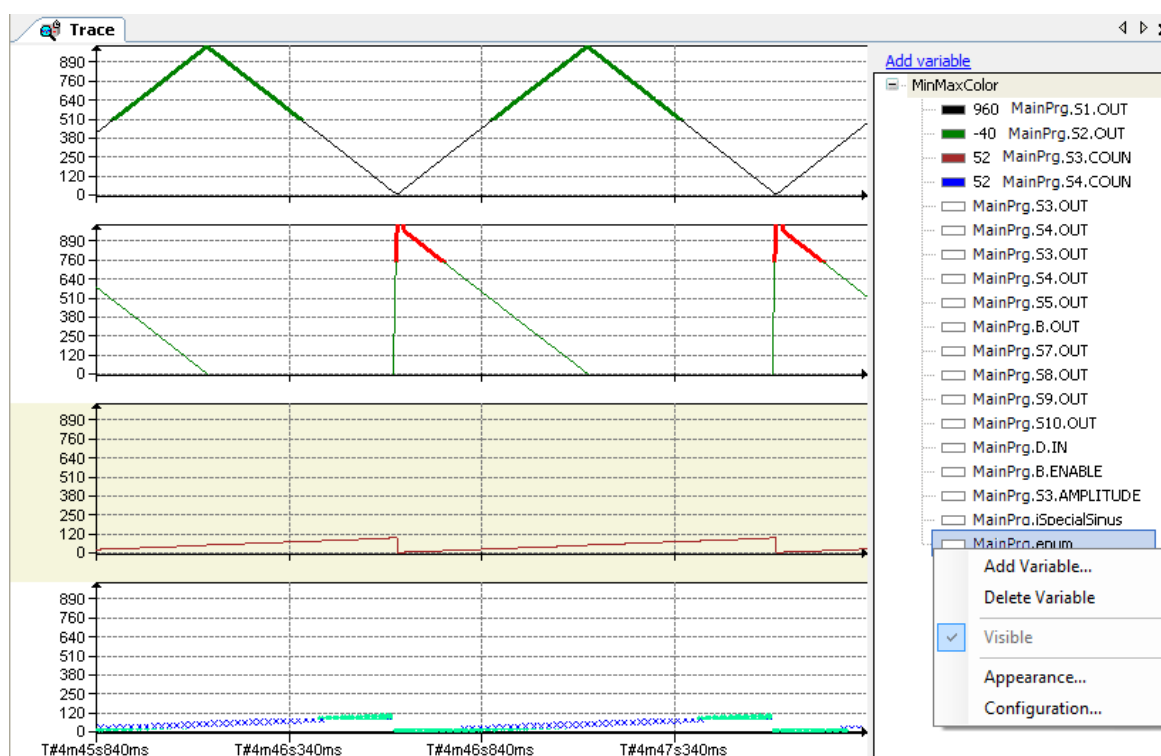


Figure 8-32. Trace Editor Window, in Online Mode

The Figure 8-32 shows an example of tracing in online mode. Four variables have been selected for display in the variables tree in the right part of the dialog. In the context menu of this part of the window see the commands available to add or delete variables, to set them visible, to change the appearance, or to reach the configuration dialog.

Commands and Zoom Functionality

For "working" in the Trace Editor in offline and in online mode there are commands available in the Trace menu as well as in the trace window. The *Trace* menu provides commands for operating the trace graph. With focus on the diagram in the trace window the context menu provides the trace menu commands too. The trace configuration with trigger setting and the variable selection can only

be done in the context menu of the right part of the *Trace* window where the trace tree is visualized. Up to the selection in the *Trace* tree different commands are enabled.

Notice that besides the zoom commands also a *Scroll Mouse* can be used to zoom the display of the coordinate system in the trace window.

Configuration of the Trace

A *Trace* object can be added in the *Devices* view window by the *Add Object* command and by *Edit Object*. Primarily the main editor window, titled with “<name of trace object>” (🔍), shows an empty area in the left part where later the trace graph will be displayed and the trace tree as configured in the *Configuration...* dialog in the right part.

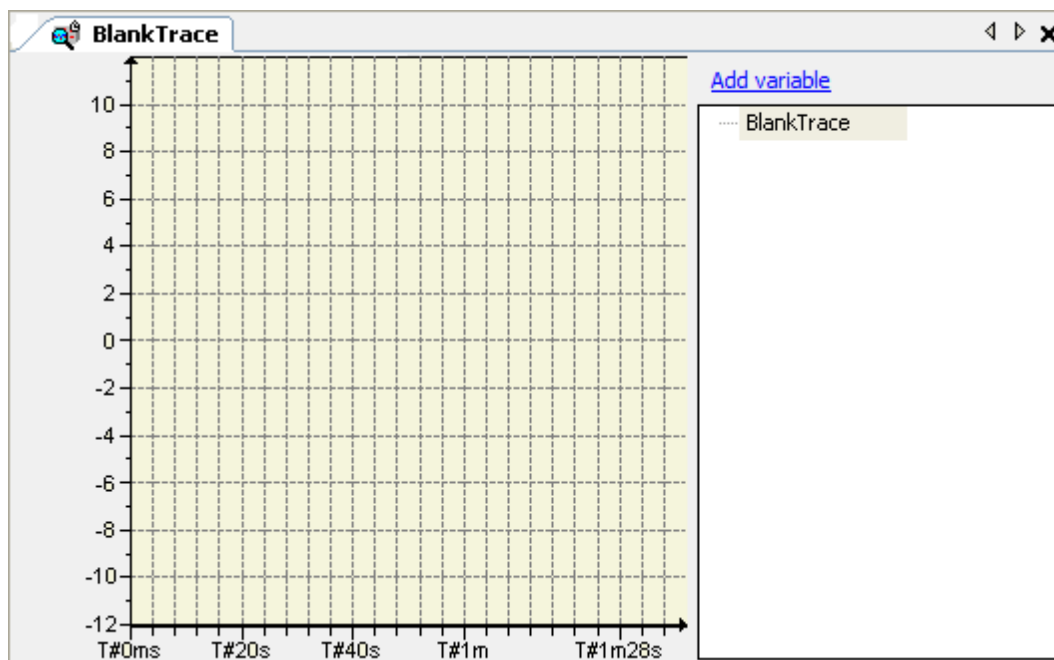


Figure 8-33. Trace Editor Window, No Configuration

A *Trace* defines one or several variables to be traced. For this common settings can be defined concerning the start of the trace, sampling, trigger conditions, buffer size, responsible task etc. Besides that, the “appearance” (display parameters for the coordinate system where the curve will be displayed) of every trace variable has to be set.

To set up a trace use the context menu in the right part of the trace window:

- *Add Variable*: dialog to add a variable and set some display parameters (color, graph type etc.)
- *Delete Variable*: to delete the selected variables.
- *Visible*: to set a selected variable visible.
- *Appearance*: dialog to determine the appearance of the graph (this command is grayed as long as no configuration is loaded).
- *Configuration*: dialog for the definition of the trace (conditions, curve type etc., and once again appearance of the graphs) with its particular trace variables.

A trace configuration can be stored to an external file *.trace and be reloaded to the trace window.

The *Trace* menu provides additional commands for working in the currently displayed trace graphs.

Add Variable

Trace configuration and variable dialog opens when using command *Add Variable* in the context menu of the trace tree window in the Trace editor.

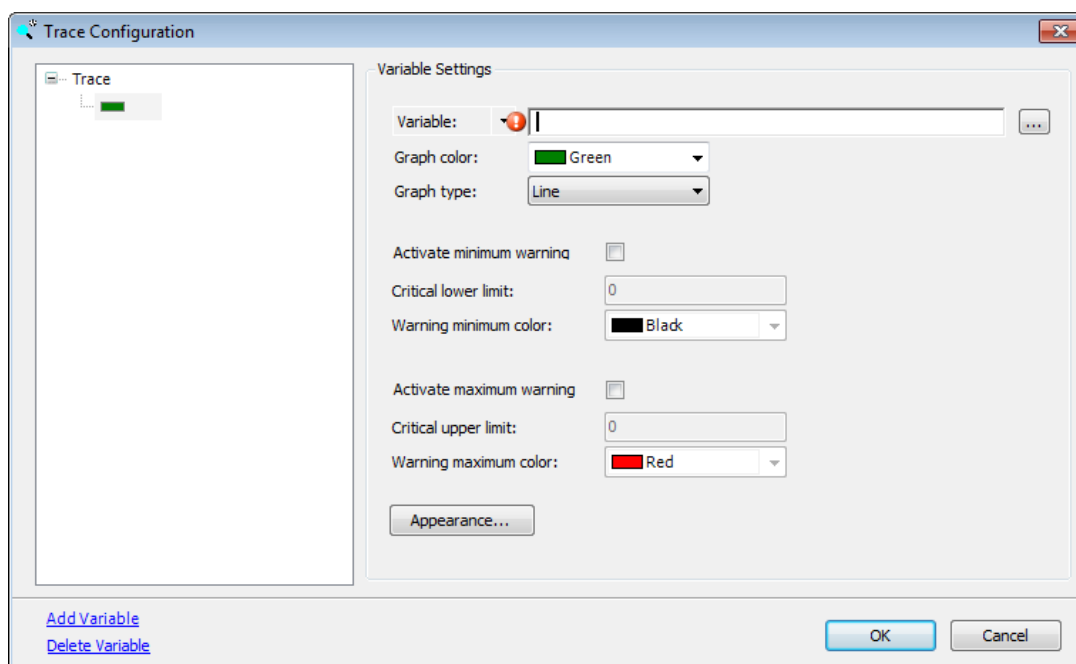


Figure 8-34. Trace Configuration Dialog

- **Variable:** Enter here the name (path) of the variable to be traced. The *Input Assistant*  can be used in order to get a valid entry. Check and edit the current configuration settings. The variable will be immediately entered in the record tree. To modify the variable settings later, just select the variable entry in the record tree again and use the *Variable Settings* dialog again.
- **Graph color:** From the given color selection list choose a color in which the trace curve for the variable should be displayed.
- **Graph type:** Specify here how the particular data points in the trace curve of the variable should be displayed. It is recommended to use the *Line* type for big volumes of data. See Table 8-6.

Type	Description
None	No graph will be displayed.
Line	The points are connected to a line.
Crosses	The points are indicated by crosses.
Step	The points are connected in shape of a staircase: vertical line up to the y-value of the next point, from there horizontal line up to the x-value.
Dots	The points are indicated by dots.

Table 8-6. Graph Type

- **Activate minimum warning:** If this option is activated, the trace graph will be displayed in the color defined in *Warning Minimum Color* as soon as the variable exceeds the value defined in *Critical lower limit*.
- **Critical lower limit:** See above, *Activate minimum warning*.
- **Warning minimum color:** See above, *Activate minimum warning*.
- **Activate maximum warning:** If this option is activated, the trace graph will be displayed in that color, which is defined in *Warning maximum color* as soon as the variable exceeds the value, defined in *Critical upper limit*.
- **Critical upper limit:** See above, *Activate maximum warning*.
- **Warning maximum color:** See above, *Activate maximum warning*.

Configuration

- Record Settings
- Variable Settings

The *Trace Configuration* dialog opens on command *Configuration...* which is available in the context menu of the trace tree on the right part of the main Trace Editor window. By double click on the trace name or the variable names the dialog opens, too. Here you configure which variables should be traced and which parameters should be set for the tracing and the defined variables.

[NOTE: The settings done in the dialog *Record Settings* are valid for all variables in the trace graph.]

The *Trace Configuration* is composed of one trace record configuration (Record Settings) and of the dedicated variable settings. The configured trace variables are visualized in the trace tree in the left part of the trace configuration dialog and is the same than the one in the main trace editor window.

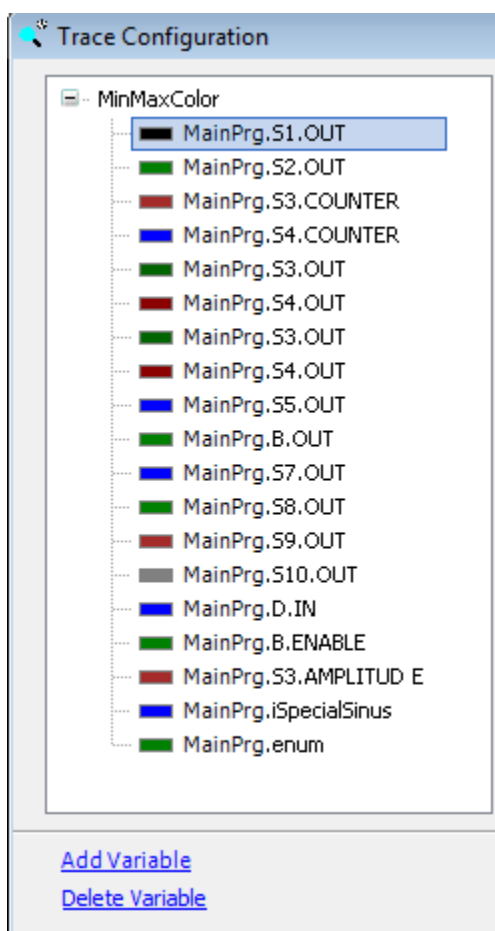


Figure 8-35. Trace Tree in the Trace Configuration Dialog

See on top level the name of the configured trace and indented below the names of the tracing variables. For changing the record settings select the name of the trace on top level of the trace tree. Then the *Record Settings* dialog is opened.

For adding a variable to the trace or deleting one see the commands below the record tree: *New Variable*, *Delete Variable*.

For changing the variable settings select the name of the desired variable, then the *Variable Settings* dialog is opened. In addition, it is possible to select several items with <SHIFT> + mouse click or <CTRL> + mouse click of the variable list. Then the changing in the *Variable Settings* dialog is effected on all of the selected items.

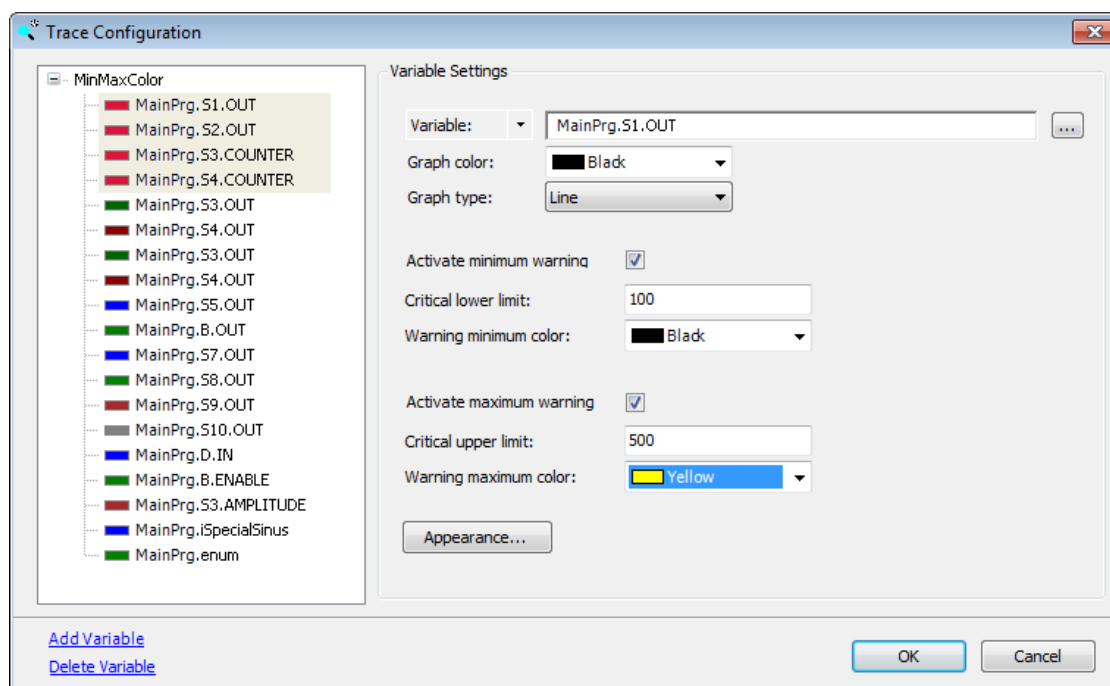


Figure 8-36. Multi Selection of Variables

With <SHIFT> or <CTRL> + mouse click multiple selection of variables is possible for changing the settings or properties in one-step.

See in the following the description of the possible Trace configuration and Variable configuration.

Record Settings

See in the following a description of the *Trace Configuration* dialog for record settings.

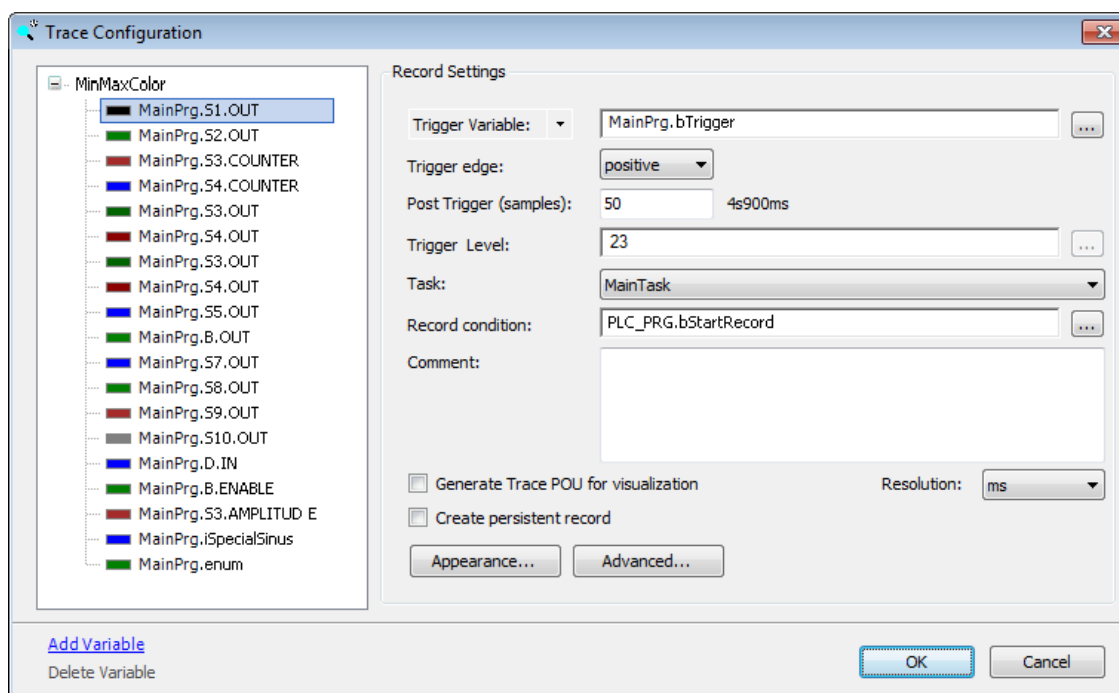


Figure 8-37. Trace Configuration, Record Settings

- **Trigger Variable:** Optionally a trigger can be configured, which should determine the time span of tracing according to certain conditions. The following settings have to be made for this purpose: Enter a boolean variable, an expression or an analogue variable. In addition, enumeration variables or property variables can be entered. When this variable has reached a defined value according to the type of *Trigger Edge*, the tracing will be terminated after having additionally sampled that percentage of values which is defined in the 'Position' field (*Pos-trigger field*). This means that as soon as the trigger gets TRUE or reaches a certain value, the trace logging will be continued for a defined number of cycles. By button  the Input Assistant can be used to get an appropriate trigger variable. Notice the possibility of controlling additionally the start of the tracing by the *Record Condition*.
- **Trigger Edge:** See Table 8-7.

Edge	Description
None	No trigger effect at all (Default)
Positive	Trigger event on rising edge of the boolean trigger variable or as soon as the value defined by 'Level' for an analogue trigger variable is reached by an ascending run.
Negative	Trigger event on falling edge of the boolean trigger variable or as soon as the value defined by 'Level' for an analogue trigger variable is reached by a descending run.
Both	Trigger event on the conditions described for 'positive' and 'negative' (see above).

Table 8-7. Trigger Edge

- **Post Trigger (samples):** sets the number of records per trace signal. They are registered after the trigger. Default value is 50, the range is from 0 through (232 – 1).
- **Trigger Level:** If an analogue variable is used as trigger variable, define here the value of this variable at which the trigger event will be caused. Enter a value directly or a variable defining the value (also ENUM constants are allowed). Default: empty.
- **Task:** From the list of available tasks select that one after the execution of which the trace variable should be read. Default: The first task in the Task Configuration tree.
- **Record condition:** A boolean variable, a value or a boolean expression can be entered here. As soon as this condition gets TRUE, the trace sampling will be started. If nothing is entered here, the trace recording will start as soon as the Trace Configuration has been downloaded and the application is running.
- **Comment:** A comment text might be entered here concerning the current record.
- **Generate Trace POU for visualization:** Activating the associated checkbox induces the implicit build of the component <Tracename>_<Taskname>_VISU. Use this option, if you want to integrate the trace readout within a visualization.
- **Appearance:** This button opens the *Edit Appearance* dialog, where you can set up the display of the trace window for the currently configured record (axes, colors and scroll behavior).
- **Advanced:** This button opens the *Trace Advanced Settings* dialog.

Trace Advanced Settings

The Trace Advanced Settings open with a click on the *Advanced* button in the screen *Trace Settings*, group *Record Settings*.

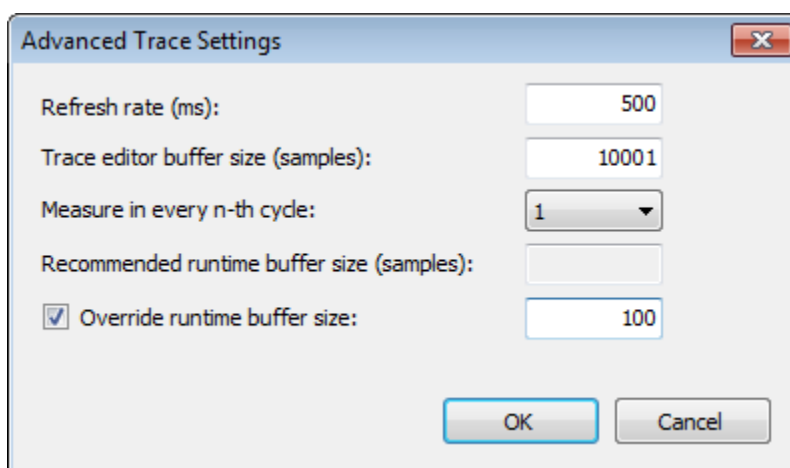


Figure 8-38 Trace Advanced Configuration

- *Refresh rate (ms)*: Time interval in milliseconds, in which the update of the trace window in online mode will be done. The value must be between 150ms and 10000ms. Default: 500ms. If you use the, the data is transferred in time-defined delivery system for the programming system. If don't, then the data are transmitted every 200 ms for the programming system
- *Trace editor buffer size (samples)*: Enter here the size of the buffer of Trace editor (write). This buffer must be greater or equal than twice the size of the runtime buffer.
- *Measure in every x-th cycle*: Define the number of cycles which should be left out before the variable should be read again. By $(\text{Buffer Size} * \text{Measure in every } x\text{-th cycle} * \text{Task interval})$ the minimum guarded time span can be estimated.
- *Recommended runtime buffer size (Samples)*: Here is shown the recommended size of runtime buffering for each monitoring signal. This value is calculated based on task cycle time, the update time and the value of every n-th cycle.
- *Override runtime buffer size*: If this option is selected, instead of the recommended value for the runtime buffer size, the value entered here will be used. The value must be at least 10 and no larger than the size of the trace editor buffer.

NOTE: If you want to display the trace curve of a trace variable in different appearances at a time, you must assign this variable to a further trace object having different appearances.

The following settings are done for the X axis and the Y axis of the trace graph and they are visible when the trace diagram is displayed in single channel view.

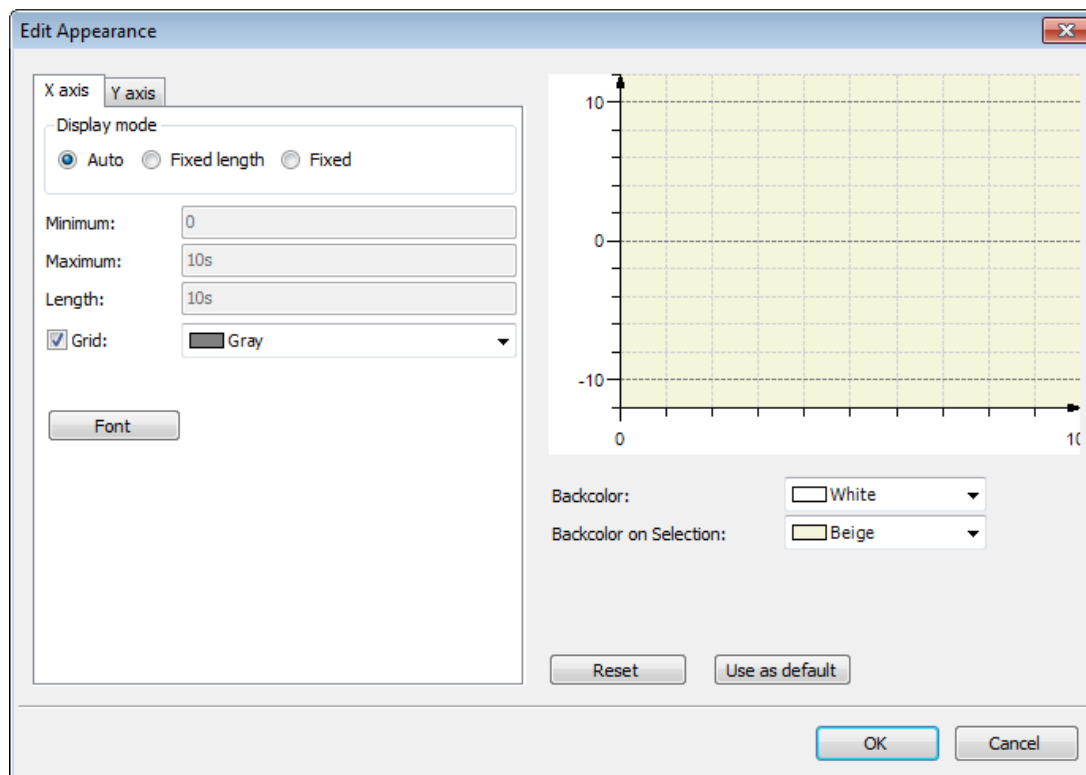


Figure 8-39. Edit Appearance Dialog

- *Auto (X axis/Y axis):* If this option is activated, the time span for tracing, that is the time span displayed on the x-axis of the trace window, will automatically be scaled according to the buffer size currently defined for the record in the runtime system.
- *Fixed Length (X axis tab):* If this option is selected, the range displayed on the time axis has a fixed length. This length must be set in the length field. The scale is also adjusted for the length and the graph scrolling is done automatically for the visible range, so will always be displayed in the chart the last generated data.
- *Fixed (X axis and Y axis tabs):* By selecting this option, the range displayed on the axis is defined by the minimum and maximum fields.
- *Minimum (X axis and Y axis tabs):* Sets the minimum value displayed on the selected axis.
- *Maximum (X axis and Y axis tabs):* This value defines the maximum value displayed on the selected axis.
- *Length (X axis tab):* Sets the length of the range displayed on the time axis.
- *Grid (X and Y axis):* If this option is activated a grid will be displayed. The major ticks will be elongated by dotted lines for this purpose. The color of the grid lines can be chosen from the color selection list.
- *Description (Y axis):* If this option is activated, the text entered in the edit field will be displayed at the upper end of the y-axis. For the x-axis no description will be displayed here.
- *Font (X and Y axis):* This button opens the standard dialog for defining the font for the trace display.
- *Backcolor:* From the given color selection list choose the background color for the coordinate system, which should be used as long as this is not selected in the trace window.
- *Backcolor on Selection:* From the given color selection list choose the background color for the coordinate system, which should be used as long as this is selected in the trace window.
- *Reset:* With this command the appearance is set to its default value.
- *Use as default:* With this command the current appearance is set as default. They are used when a new trace or variable is configured.

Variable Settings

See in the following a description of the trace configuration dialog for variable settings.

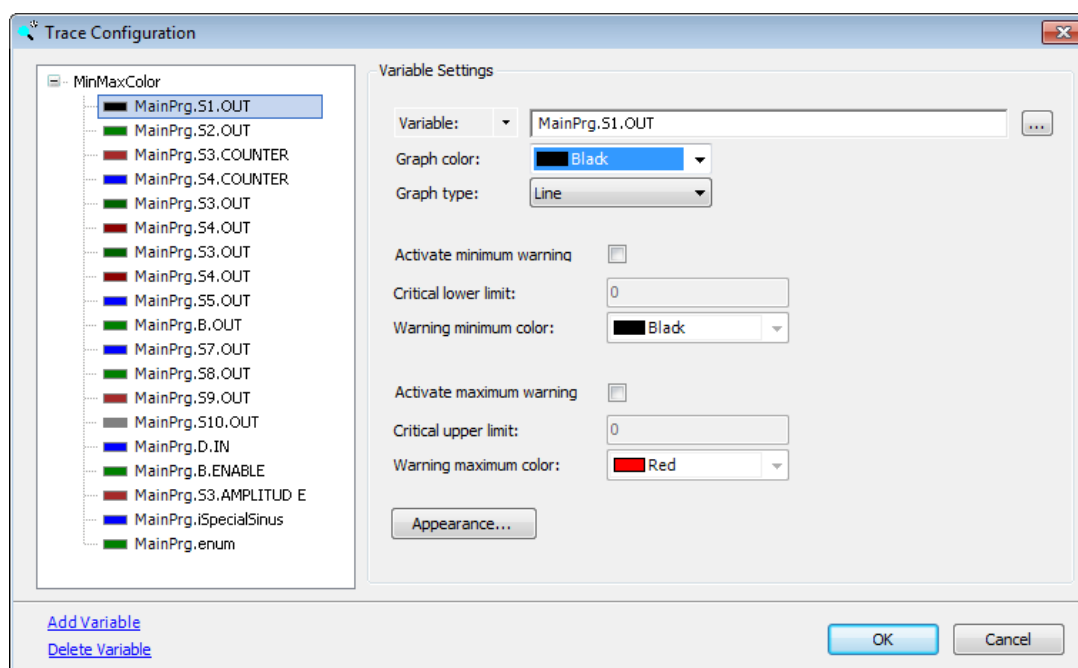


Figure 8-40. Trace Configuration, Variable Settings

For the trace record you must define the variables to be "traced" and to be displayed in the record coordinate system. In order to add a variable to the record use button *Add Variable* (bottom part of the window) in the left part of the *Trace Configuration* window. In the right part of the window now the *Variable Settings* dialog will appear and at the same time an empty variable entry will be added in the record tree. Define the variable:

- **Variable:** Enter here the name (path) of the variable to be traced. Beside of data variables it is possible to trace the content of pointers, properties, references or ARRAY elements. The *Input Assistant* (☰) can be used in order to get a valid entry. Check and edit the current configuration settings (see below). The variable will be immediately entered in the trace tree. To modify the variable settings later, just select the variable entry in the trace tree and use the *Variable Settings* dialog again.
- **Graph color:** From the given color selection list choose a color in which the trace curve for the variable should be displayed.
- **Graph type:** Specify here how the particular data points in the trace curve of the variable should be displayed. It is recommended to use the *Line* type for big volumes of data. Check Table 8-8 for all possible types of graphics.

Type	Description
None	No graph will be displayed.
Line	The points are connected to a line.
Crosses	The points are indicated by crosses.
Spline	A spline graph created of the recorded values will be displayed.
Step	The points are connected in shape of a staircase: vertical line up to the y-value of the next point, from there horizontal line up to the x-value.
Dots	The points are indicated by dots.

Table 8-8. Types of Graphics

- *Activate minimum warning*: If this option is activated, the trace graph will be displayed in the color defined in *Warning minimum color* as soon as the variable exceeds the value defined in *Critical lower limit*.
- *Critical lower limit*: See above, *Activate Minimum Warning*.
- *Warning minimum color*: See above, *Activate Minimum Warning*.
- *Activate maximum warning*: If this option is activated, the trace graph will be displayed in that color, which is defined in *Warning maximum color* as soon as the variable exceeds the value, defined in *Critical upper limit*.
- *Critical upper limit*: See above, *Activate maximum warning*.
- *Warning maximum color*: See above, *Activate maximum warning*.
- *Appearance*: This button opens the *Appearance of the Y-axis* dialog, where you can set up the display of the trace window for the currently configured Y-axis (colors and scroll behavior) for every variable in its own style. These settings are used when the trace diagram is displayed in multi-channel view.

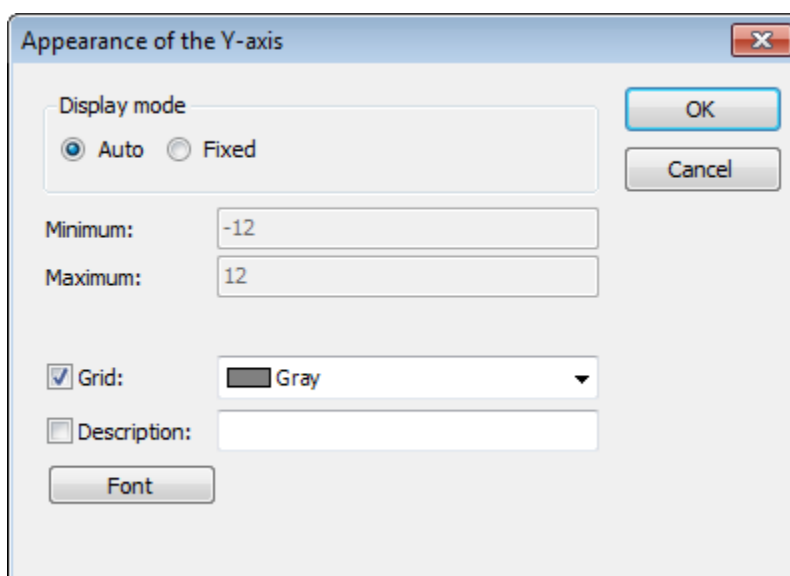


Figure 8-41. Edit Appearance of the Y-Axis Dialog

- *Auto*: With this option selected, the axis is automatically scaled according to the trace editor's buffer contents.
- *Fixed*: By selecting this option, the range displayed on the axis is defined by the minimum and maximum fields.
- *Minimum*: Sets the minimum value displayed on the selected axis.
- *Maximum*: This value defines the maximum value displayed on the selected axis.
- *Grid*: If this option is activated a grid will be displayed. The major ticks will be elongated by dotted lines for this purpose. The color of the grid lines can be chosen from the color selection list.
- *Description*: If this option is activated, the text entered in the edit field will be displayed at the upper end of the y-axis. For the x-axis no description will be displayed.
- *Font*: this button opens the standard dialog for defining the font for the Trace display.

Appearance

You find this command in the context menu of the trace tree in the right part of the main window of the Trace-Editor. It opens the *Edit Appearance* dialog in the Trace editor. The dialog is available, when the name of the trace on top level of the trace tree is selected. If a variable in the trace tree is selected, the *Appearance of the Y-axis* dialog opens.

The dialog is also available, when you are in the *Configuration* dialog of the trace.

In both dialogs you define how the trace data will be displayed in the trace diagram. That is you define the looking of the coordinate system (axes configuration, colors, grid etc.) in which the values' graphs of the variables will be displayed. The appearance settings of the record (*Edit Appearance* dialog) are used in the default single channel mode, but the appearance settings of the variables (Appearance of the Y-axis dialog) are visible in the multi-channel mode.

Trace Editor in Online Mode

Overview

The "Trace" is running like an application on the current device. You get indicated an activated Trace sampling by the green color of the Trace object in the devices tree.

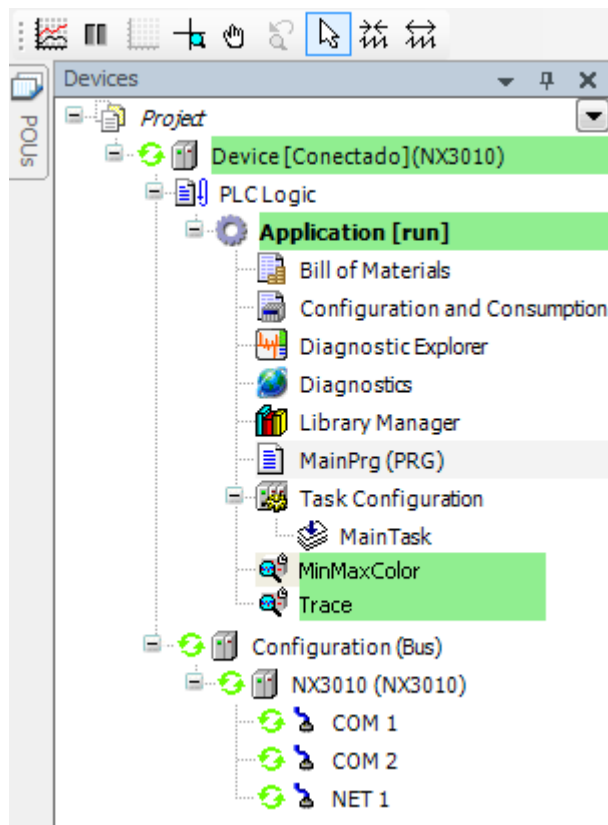


Figure 8-42. Active Trace in Devices Tree

Downloaded Trace

For the first use and later after any changes in the trace configuration you have explicitly to download the Trace application in order to activate the tracing for the currently running application. After having changed the application the trace is terminated automatically and you have to re-download it. After having logged out and logged in from/to the application without having made any changes, the trace will continue without a re-download.

Configuration of Trace Graphs

The *Configuration* dialog for the variables is also available in online mode and changes can be done online. You can double-click the name of the variable in the configuration tree in the right part of the trace editor or you can open the context menu there and select the configuration command.

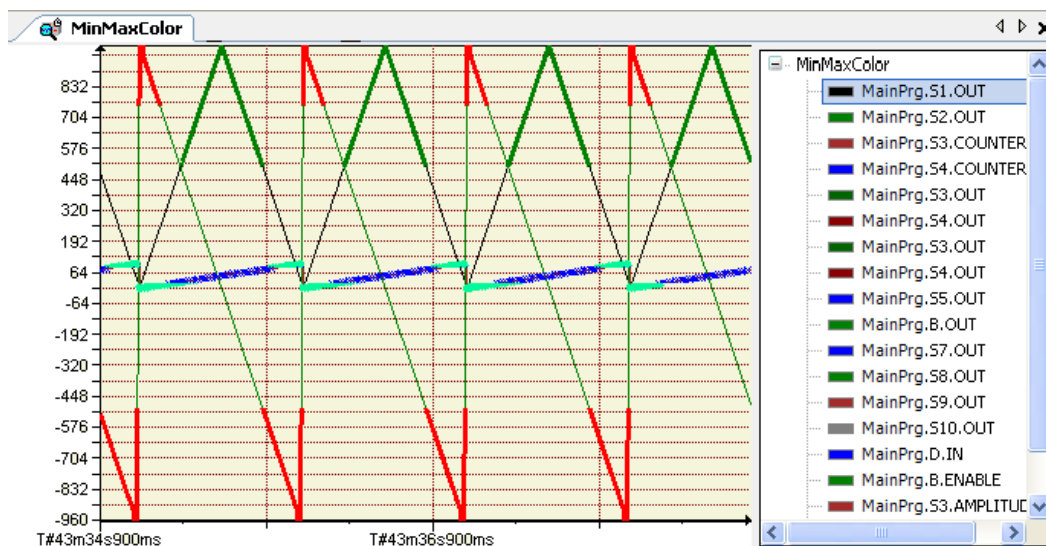


Figure 8-43. Trace Window in Online Mode, Example in Single Channel Mode

Control the Trace Display

By default the commands of the Trace menu are available for controlling the currently viewed trace graph (*Start/Stop Trace*, *Reset*) and for modifying the currently displayed section of the trace curves (mouse scrolling, mouse zooming, etc.).

The value range of the trace graphs depends on the specific settings for the trace. However, you can manipulate it manually by the scroll and zoom functions available in the Trace menu. This can be done with shortcuts, too. For vertical scrolling of the current view of the trace along the y-axis use the Up arrow key to move up and the Down arrow key to move down. For horizontal scrolling along the x-Axis use the Left or right arrow key while pressing the <ALT> key.

If the *Multi-Channel* option is enabled, every graph has its own diagram and a selected graph can be scrolled as the only one alone with the shortcuts above. To scroll all the graphs simultaneously use the Arrow keys with an additional pressed <CTRL> key to move the y-axis.

All the trace shortcuts listed in Shortcuts for trace.

Notice in this context also the zoom functionality of a **Mouse Scrolling**.

When the cursor is placed on the trace window the respective value of the x-axis is being showed in the status line.

Watch List Editor

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.

Create Watchlist

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.MainPrg.ivar” (as shown in Figure 8-44).

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.

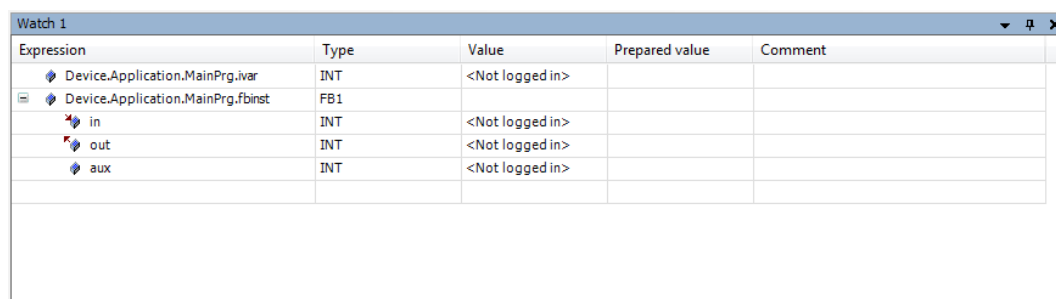
After having closed the edit frame the *Type* will be added automatically in column Type.

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.MainPrg.fbinst”). They can be displayed/hidden in a fold via a mouse action on the plus-/minus sign.



Expression	Type	Value	Prepared value	Comment
 Device.Application.MainPrg.ivar	INT	<Not logged in>		
  Device.Application.MainPrg.fbinst	FB1			
 in	INT	<Not logged in>		
 out	INT	<Not logged in>		
 aux	INT	<Not logged in>		

Figure 8-44. Watch View in Offline Mode

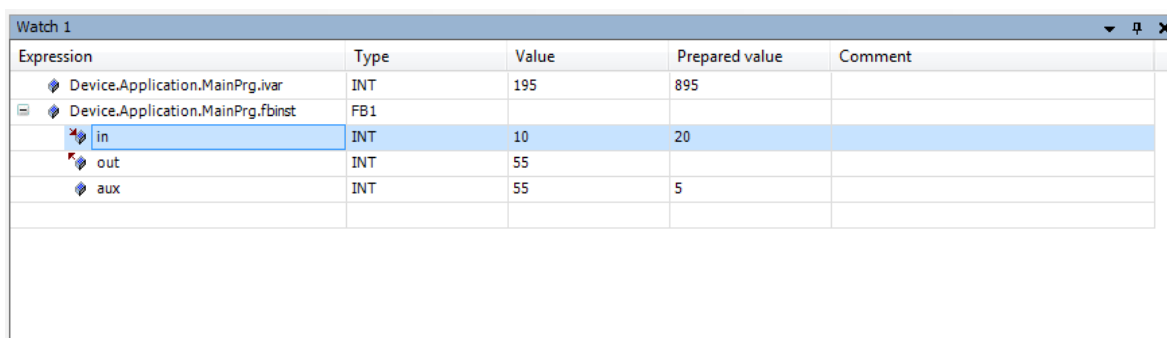
In online mode the view can be used for monitoring.

Watch List in Online Mode

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 Watchlist**.



Expression	Type	Value	Prepared value	Comment
Device.Application.MainPrg.iavar	INT	195	895	
Device.Application.MainPrg.fbinst	FB1			
in	INT	10	20	
out	INT	55		
aux	INT	55	5	

Figure 8-45. Watch View in Online Mode

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.

Write Values

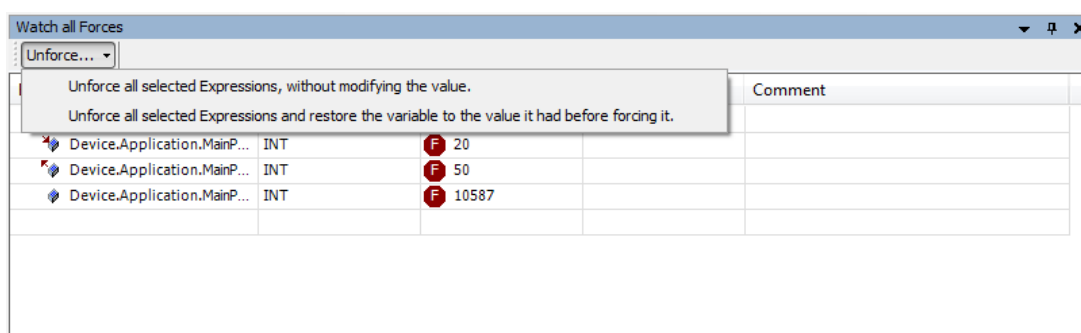
In column *Prepared Value* you can enter a desired value which will be written or forced to the respective expression on the controller by command *Write Values* or *Force Values* like usable in other monitoring views (for example declaration editor).

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.*



Expression	Type	Value	Prepared value	Comment
Device.Application.MainP...	INT	F 20		
Device.Application.MainP...	INT	F 50		
Device.Application.MainP...	INT	F 10587		

Figure 8-46. Watch All Forces

MODBUS Editor

To add a MODBUS device, just click with the right mouse button on the communication port that you want to enable the Protocol and choose the option *Add Device...*

A screen will open where the MODBUS are available for the selected communication port, now simply choose the desired option and click *Add*.

The MODBUS for symbolic mapping devices are available from the version 1.40 of MasterTool IEC XE to later. In order to use them properly the Nexto UCP version 1.3.0.20 or later is necessary.

MODBUS RTU Master for Direct Representation (%Q)

By adding the MODBUS RTU Master for Direct Representation (%Q), the following screen will appear.

MODBUS Master Configuration

Master Settings

%Q Start Address of Diagnostics Area
22040

Size
20

Advanced...

Used range: %QB22040..%QB22059

Devices

Instance Name	Slave Address

Add... Remove Edit...

Data Type	Read Data Start Address	Read Data Size	Read IEC Variable	Write Data Start Address	Write Data Size	Write IEC Variable

Add... Remove Edit...

Figure 8-47. MODBUS RTU Master for Direct Representation Screen

Configuration	Description	Default	Possibilities
%Q Start Address of Diagnostics Area	Initial address for the diagnosis variables	-	Any address of %Q area, limited according to CPU used
Size	Diagnosis Area Size	20	20

Table 8-9. Initial Address for Diagnosis on MODBUS Master for Direct Representation

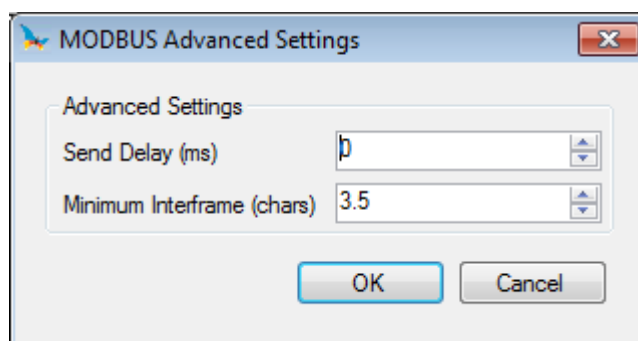


Figure 8-48. MODBUS Master for Direct Representation Advanced Settings

Configuration	Description	Default	Possibilities
Send Delay (ms):	Delay for the answer transmission	0	Any address of %Q area, limited according to CPU used
Minimum Interframe (chars):	Minimum silence time between different frames	3.5	3.5 to 100.0

Table 8-10. Timings Setting on MODBUS Master for Direct Representation

Adding a Device

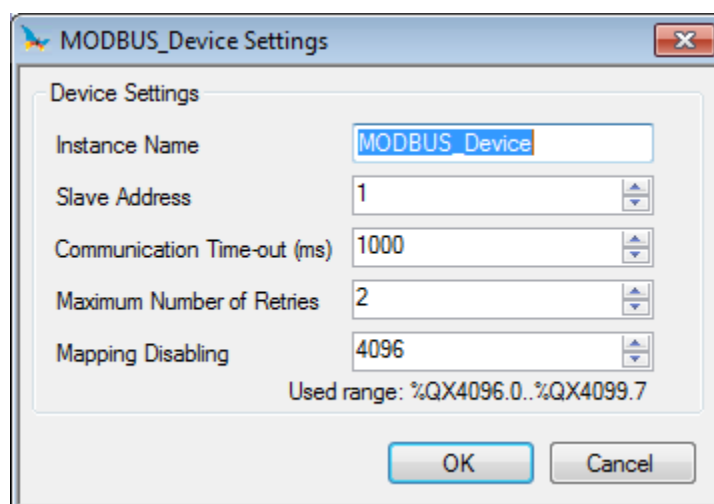


Figure 8-49. Adding a Device on MODBUS Master for Direct Representation

Configuration	Description	Default	Possibilities
Instance Name	Instance name	MODBUS Device	Identifier, as IEC 61131-3
Slave Address	MODBUS slave address	1	0 to 255
Communication Time-out (ms)	Defines the application level time-out	1000	10 to 65535
Maximum Number of Retries	Defines the numbers of retries before reporting a communication error	0	0 to 9

Mapping Disabling	Initial address used to disable the MODBUS relations configured in Devices Mapping field (%QX.X)	0	Any address of %Q area, limited by the used CPU

Table 8-11. MODBUS Device Configurations on MODBUS RTU Master for Direct Representation

Adding a MODBUS Relation

After inserting a device, the button for adding a MODBUS relation will be enabled.

Click on *Add...* button and a screen will pop up, where the user can choose the relation data type. Regardless of the data chosen, the following screen will open where the appropriate settings should be done.

Figure 8-50. MODBUS Relation

Configuration	Description	Default	Possibilities
Function	MODBUS function type	Read	Read Write Read / Write Mask Write
Polling (ms)	Communication Period (ms)	100	0 to 3600000
Mapping Diagnostic Area	Initial address for the MODBUS relation diagnostics	-	Any address of %Q, limited according to the used CPU
Read Data Start Address	Initial address for the MODBUS reading data	0	0 to 65535
Read Data Size	Data number for the MODBUS reading	8	Depends on the function used
Read IEC Variable	Initial address for the reading variables (%I)	1	Any address of %Q, limited according to the used CPU

Written Data Start Address	Initial address for the MODBUS writing data	0	0 to 65535
Written Data Size	Data number for the MODBUS writing	8	Depends on the function used
Written IEC Variable	Initial address for the writing variables (%Q)	1	Any address of %Q, limited according to the used CPU
Mask IEC Variable	Variables initial address for the writing mask (%Q)	1	Any address of %Q, limited according to the used CPU

Table 8-12. MODBUS Master for Direct Representation Relation

MODBUS RTU Master for Symbolic Mapping

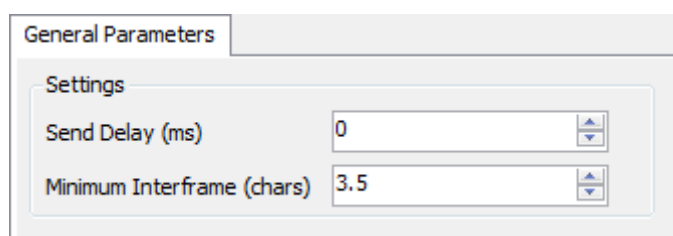


Figure 8-51. MODBUS RTU Master for Symbolic Mapping Screen

Configuration	Description	Default	Possibilities
Send Delay (ms)	Delay for the answer transmission	0	0 a 65535
Minimum Interframe (chars)	Minimum silence time between different frames	3.5	3.5 a 100.0

Table 8-13. Timing Configurations on MODBUS RTU Master for Symbolic Mapping

Adding a Device

To add a device into the configuration of the MODBUS RTU Master for Symbolic Mapping use the MODBUS RTU Master context menu.

General Parameters

The configuration of the slave devices, seen on Figure 8-52, follow the parameters below:

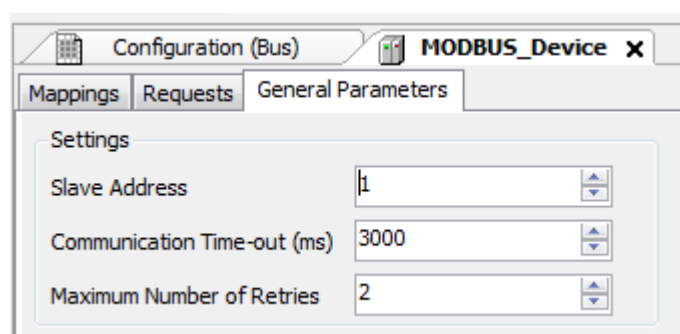


Figure 8-52. Device General Parameters on MODBUS RTU Master for Symbolic Mapping

Configuration	Description	Default	Possibilities
Slave Address	MODBUS slave address	1	0 to 255
Communication Time-out (ms)	Defines the application level time-out	3000	10 a 65535
Maximum Number of Retries	Defines the numbers of retries before reporting a communication error	2	0 to 9

Table 8-14. Device Configurations on MODBUS RTU Master for Symbolic Mapping

Mapping Configuration

The mapping configuration on MODBUS RTU Master for Symbolic Mapping, seen on Figure 8-53, follow the parameters described on Table 8-15:

Figure 8-53. MODBUS RTU Master for Symbolic Mapping Screen

Configuration	Description	Default	Possibilities
Variable Value	Name of the symbolic variable	-	Variable name declared in a program or GVL
Data Type	MODBUS data type	-	Coil Writing (1 bit) Coil Reading (1 bit) Holding Register Writing (16 bits) Holding Register Reading (16 bits) Holding Register – Mask AND (16 bits) Holding Register – Mask OR (16 bits) Input Register (16 bits) Input Status (1 bit)
Data Start Address	MODBUS data initial address	-	1 to 65536
Data Size	MODBUS data size	-	1 to 65536
Data Range	Range of addresses of the configured data	-	-

Table 8-15 Mapping Configurations on MODBUS RTU Master for Symbolic Mapping

Request Configuration

The request configuration on MODBUS RTU Master for Symbolic Mapping, seen on Figure 8-54, follow the parameters described on Table 8-16:

The screenshot shows a software interface for configuring MODBUS RTU Master requests. It features a 'General Parameters' tab with a table for defining request parameters. The table includes fields for Function Code, Polling (ms), Read Data Start Address, Read Data Size, Read Data Range, Write Data Start Address, Write Data Size, Write Data Range, Diagnostic Variable, and Disabling Variable. A single configuration row is shown, marked with an asterisk (*). Below the table, there are buttons to 'Generate Diagnostics Variables' and 'Generate Disabling Variables'.

Figure 8-54. MODBUS RTU Master for Symbolic Mapping Data Request Screen

Configuration	Description	Default	Possibilities
Function Code	MODBUS function type	-	01 – Coils Reading 02 – Input Status Reading 03 – Holding Registers Reading 04 – Input Registers Reading 05 – Coil Writing 06 – Register Writing 15 – Multiple Coils Writing 16 – Multiple Registers Writing 22 – Register Mask Writing 23 – Multiple Register Reading / Writing
Polling (ms)	Communication period (ms)	100	0 to 3600000
Read Data Start Address	Initial address for the MODBUS reading data	-	1 to 65536
Read Data Size	Data size for the MODBUS reading	-	Depends on the function used
Read Data Range	Range of addresses for the MODBUS data reading	-	0 to 2147483646
Write Data Start Address	Initial address for the MODBUS writing data	-	1 to 65536
Write Data Size	Data size for the MODBUS writing	-	Depends on the function used
Write Data Range	Range of addresses for the MODBUS data MODBUS writing	-	0 to 2147483647
Diagnostic Variable	Diagnostic variable name	-	Name of a variable declared on a program or GVL
Disabling Variable	Initial Variable used to disable the MODBUS relations	-	Field designed for the symbolic variable, which is used to individually disable the configured MODBUS requests. This variable may be of BOOL type. The variable may be single or array element and can also be in structures.

Table 8-16. MODBUS RTU Master for Symbolic Mapping Relation Configuration

MODBUS RTU Slave for Direct Representation (%Q)

By adding the MODBUS RTU Slave for Direct Representation, the following screen will appear

MODBUS Slave Configuration

Slave Settings

%Q Start Address of Diagnostics Area: 21695
 SlaveAddress: 1
 Size: 20
 Mapping Disabling: 4100
 Used range: %QB21695..%QB21714 Used range: %QX4100.0..%QX4103.7

Slave Mappings

	Data Type	Data Start Address	Data Size	IEC Variable
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Add... Remove Edit...

Figure 8-55. MODBUS RTU Slave for Direct Representation

Configuration	Description	Default	Possibilities
%Q Start Address of Diagnostics Area	Diagnostics variables initial address (%Q)	-	Any address area %Q, limited according to the CPU used
Size	Diagnostics area size	20	20
Slave Address	MODBUS slave address	1	1 to 255
Mapping Disabling	Initial address used to disable the MODBUS relations which are set on Slave Mappings (%QX.X)	0	Any address area %Q, limited according to the CPU used

Table 8-17. MODBUS RTU Slave for Direct Representation Configuration

Advanced Configurations

This configuration is the same as the MODBUS RTU Master for Direct Representation one.

Adding a MODBUS Relation

Click on *Add...* button and a screen will pop up, where the user can choose the relation data type. Regardless of the data chosen, the following screen will open where the appropriate settings should be done.

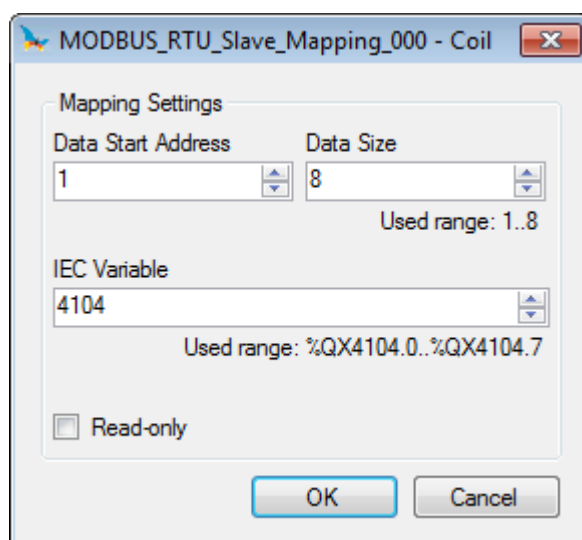


Figure 8-56. Adding a Relation on MODBUS RTU Slave for Direct Representation

Configuration	Description	Default	Possibilities
Data Type	MODBUS data type	Coil	Coil (1 bit) Holding Register (16 bits) Input Status (1 bit) Input Register (16 bits)
Data Start Address	MODBUS data initial address	0	Any address area %Q, limited according to the CPU used
Data Size	MODBUS data quantity	8	1 to 65535
IEC Variable	Variables initial address (%Q)	0	Any address area %Q, limited according to the CPU used

Table 8-18. MODBUS RTU Slave for Direct Representation Relation Configuration

MODBUS RTU Slave for Symbolic Mapping

By adding a MODBUS RTU Slave for Direct Mapping, its configuration screen is showed.

General Parameters

The configuration of the slave devices, seen on the initial screen of the MODBUS RTU Slave for Symbolic Mapping protocol are described on Figure 8-57.

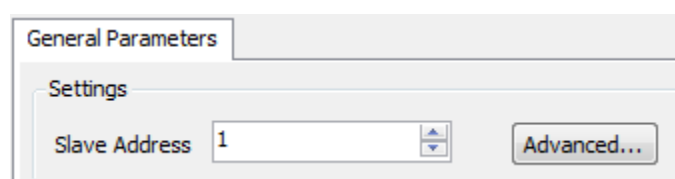


Figure 8-57. MODBUS RTU Slave for Symbolic Mapping Configuration Screen

Configuration	Description	Default	Possibilities
Slave Address	MODBUS Slave address	1	1 to 255

Table 8-19. MODBUS RTU Slave for Symbolic Mapping Configurations

The MODBUS RTU Slave for Symbolic Mapping communication times, found in the *Advanced...* button on the configuration initial screen, are divided in: Cycle Task, Send Delay and Minimum Interframe, as showed on Figure 8-58 and Table 8-20.

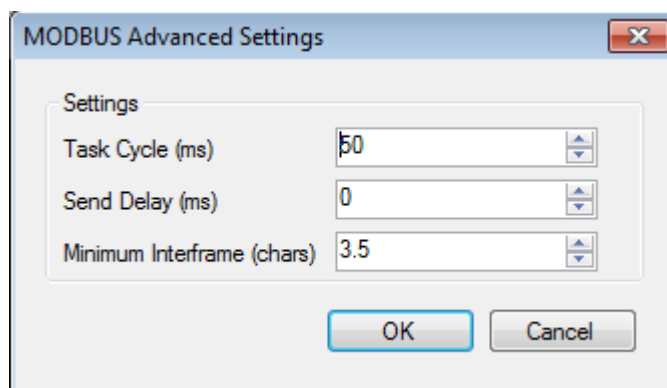


Figure 8-58. MODUS RTU Slave for Symbolic Mapping Advanced Configurations Screen

Configuration	Description	Default	Possibilities
Task Cycle (ms)	Time for the instance execution within the cycle, without considering its own execution time	50	20 to 100
Send Delay (ms)	Delay for the answer transmission	0	0 to 65535
Minimum Interframe (chars)	Minimum silence time between different frames	3.5	3.5 to 100.0

Table 8-20. MODUS RTU Slave for Symbolic Mapping Advanced Configurations

Mapping Configuration

The mapping configuration of MODUS RTU Slave for Symbolic Mapping, seen on Figure 8-59, follow the parameters described on Table 8-21:

	Value Variable	Data Type	Data Start Address	Data Size	Data Range
*					

Figure 8-59. Mapping Screen of MODUS RTU Slave for Symbolic Mapping

Configuration	Description	Default	Possibilities
Variable Value	Name of the symbolic variable	-	Name of a variable declared in a program or GVL
Data Type	MODBUS data type	-	Coil (1 bit) Input Status (1 bit) Holding Register (16 bits) Input Register (16 bits)
Data Start Address	MODBUS data initial address	-	1 to 65536
Data Size	MODBUS data quantity	-	1 to 65536
Data Range	Range of addresses of the configured data	-	-

Table 8-21. Mapping Configuration on MODUS RTU Slave for Symbolic Mapping

MODBUS Ethernet Client for Direct Representation (%Q)

By adding the MODBUS RTU Ethernet Client for Direct Representation protocol, the following screen will be shown.

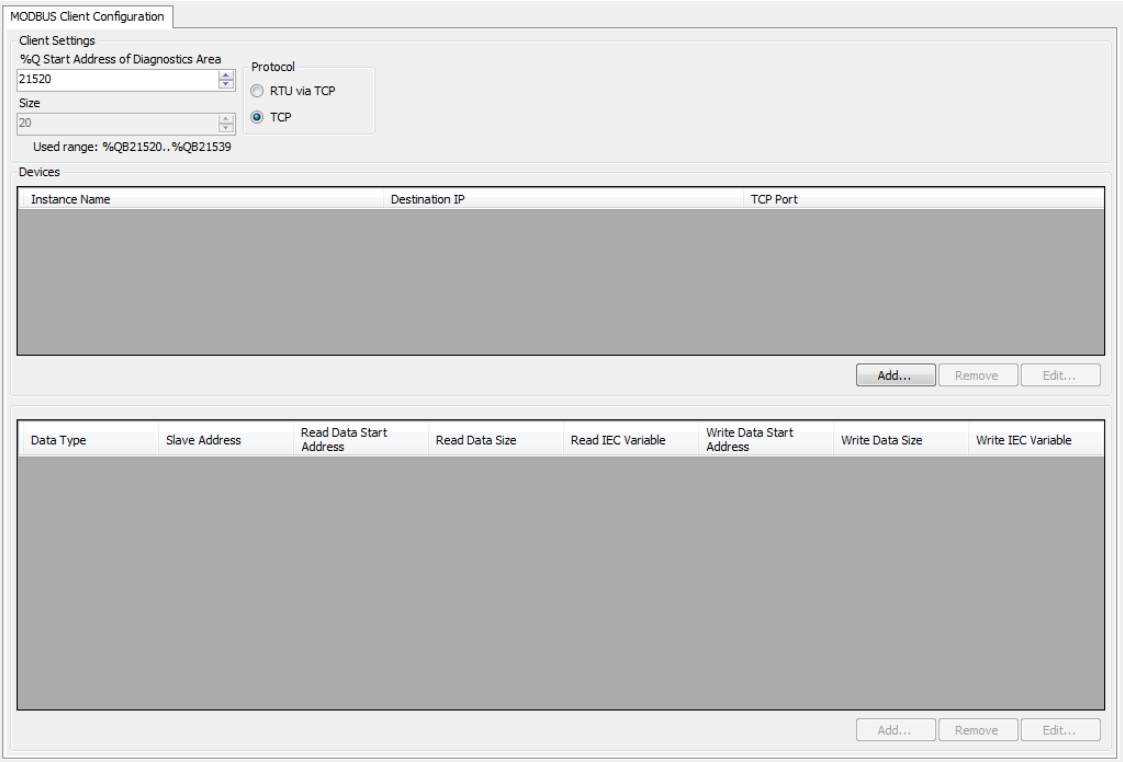


Figure 8-60. MODBUS Ethernet Client for Direct Representation

Configuration	Description	Default	Possibilities
Protocol	Protocol selection	TCP	RTU via TCP TCP
%Q Start Address of Diagnostic Area	Diagnosis variables initial address (%Q)	-	Any address of %Q area, limited according to CPU used
Size	Diagnosis area size	20	20

Table 8-22. Diagnostic Initial Address

Adding a Device

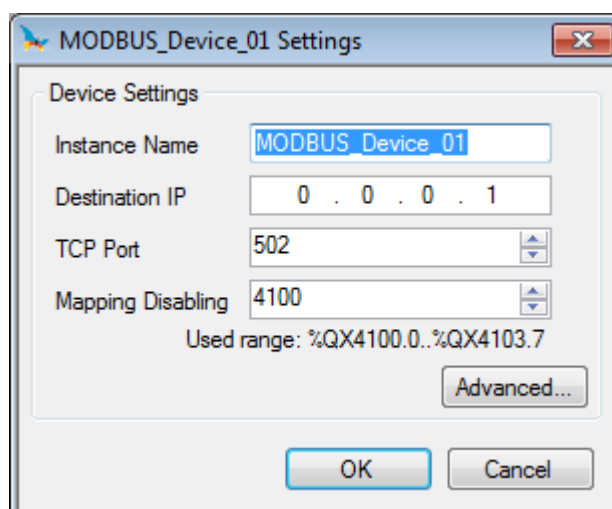


Figure 8-61. Adding a Device on MODBUS Ethernet Client for Direct Representation

Configuration	Description	Default	Possibilities
Instance Name	Instance name	MODBUS Device	Identifier, as IEC 61131-3
Destination IP	Server IP address	0.0.0.0	0.0.0.0 to 255.255.255.254
TCP Port	TCP port	502	0 to 65535
Mapping Disabling	Initial address used to disable the MODBUS relations configured in the Device Mappings field (%QX.X)	0	Any address of %Q area, limited according to CPU used

Table 8-23. MODBUS Ethernet Client for Direct Representation Configuration

Advanced Configurations

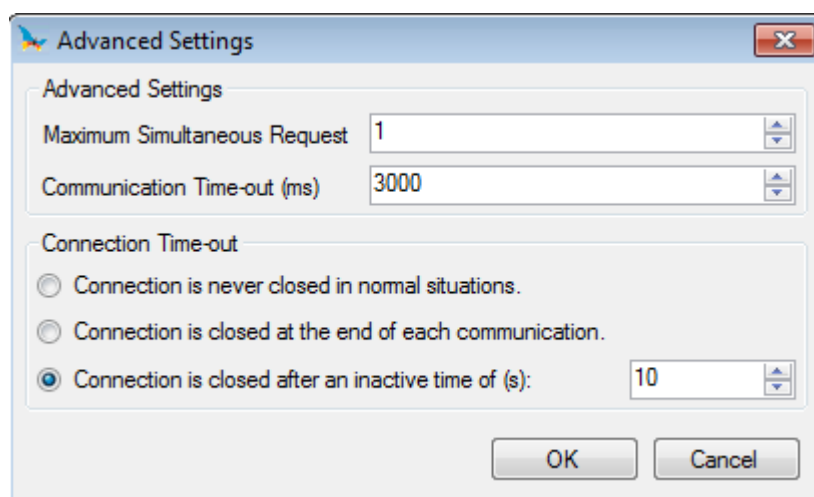


Figure 8-62. MODBUS Ethernet Client for Direct Representation Advanced Configurations Screen

Configuration	Description	Default	Possibilities
Maximum Number of Simultaneous Requests	Amount of requests handled in a same task execution cycle	1	1 to 8
Communication Timeout (ms)	Timeout TCP/IP	3000	10 to 65535
The connection is never closed in normal situations	-	-	-
The connection is closed at the end of each communication	-	-	-
The connection is closed after a time of inactivity (s)	Time without activity on a TCP/IP connection after which the connection is terminated	10	10 to 3600

Table 8-24. MODBUS Ethernet Client for Direct Representation Advanced Configurations

Adding a MODBUS Relation

This configuration is the same as the MODBUS RTU Master for Direct Representation one.

MODBUS Ethernet Client for Symbolic Mapping

The general parameters found on the initial screen of the MODBUS Ethernet Client for Symbolic Mapping (Figure 8-63), are defined as:

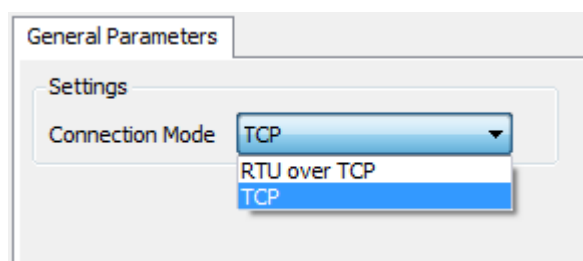


Figure 8-63. MODBUS Ethernet Client for Symbolic Mapping General Parameters Screen

Configuration	Description	Default	Possibilities
Connection Mode	Protocol selection	TCP	RTU via TCP TCP

Table 8-25. MODBUS Ethernet Client for Symbolic Mapping General Configuration

General Parameters

The configuration of the slave devices, seen on Figure 8-64, follow the parameters above:

Figure 8-64. General Parameters on MODBUS Ethernet Client for Symbolic Mapping

Configuration	Description	Default	Possibilities
IP Address	Server IP address	0. 0. 0. 0	0.0.0.1 to 255.255.255.254
TCP Port	TCP Port	502	2 a 65534
Slave Address	MODBUS Slave address	1	0 a 255

Table 8-26. MODBUS Ethernet Client for Symbolic Mapping General Configuration

Configuration	Description	Default	Possibilities
Maximum Number of Simultaneous Requests	Amount of simultaneous requests that the client can ask to the server	1	1 to 8
Communication Timeout	Time-out of the application level in ms	3000	10 to 65535
Mode	Define when the connection with the server is closed by the client	The connection is closed after a time of inactivity (s): 10 to 3600	The connection is never closed in normal situations The connection is closed at the end of each communication The connection is closed after a time of inactivity (s): 10 to 3600.
Time of Inactivity	Time of inactivity	10	3600

Table 8-27. MODBUS Ethernet Client for Symbolic Mapping Advanced Configurations

Mapping Configuration

The MODBUS Ethernet Client for Symbolic Mapping configuration, seen on Figure 8-65, follow the parameters described on Table 8-28:

	Value Variable	Data Type	Data Start Address	Data Size	Data Range
*					

Figure 8-65. MODBUS Ethernet Client for Symbolic Mapping Screen

Configuration	Description	Default	Possibilities
Variable Value	Name of the symbolic variable	-	Variable name declared in a program or GVL
Data Type	MODBUS data type	-	Coil Writing (1 bit) Coil Reading (1 bit) Holding Register Writing (16 bits) Holding Register Reading (16 bits) Holding Register – Mask AND (16 bits) Holding Register – Mask OR (16 bits) Input Register (16 bits) Input Status (1 bit)
Data Start Address	MODBUS data initial address	-	1 to 65536
Data Size	MODBUS data size	-	1 to 65536
Data Range	Range of addresses of the configured data	-	-

Table 8-28. Mappings on MODBUS Ethernet Client for Symbolic Mapping

Requests Configuration

The MODBUS Ethernet Client for Symbolic Mapping request configuration, seen on Figure 8-66, follow the parameters described on Table 8-29:

Mappings Requests General Parameters

	Function Code	Polling (ms)	Read Data Start Address	Read Data Size	Read Data Range	Write Data Start Address	Write Data Size	Write Data Range	Diagnostic Variable	Disabling Variable
*										

Diagnostics Variable Type: NXMODBUS_DIAGNOSTIC_STRUCTS.T_DIAG_MODBUS_ETH_MAPPING_1

Generate Diagnostics Variables Generate Disabling Variables

Figure 8-66 MODBUS Ethernet Client for Symbolic Mapping Requests Screen

Configuration	Description	Default	Possibilities
Function Code	MODBUS function type	-	01 – Coils Reading 02 – Input Status Reading 03 – Holding Registers Reading 04 – Input Registers Reading 05 – Coil Writing 06 – Register Writing 15 – Multiple Coils Writing 16 – Multiple Registers Writing 22 – Register Mask Writing 23 – Multiple Register Reading / Writing
Polling (ms)	Communication period (ms)	100	0 to 3600000
Read Data Start Address	Initial address for the MODBUS reading data	-	1 to 65536
Read Data Size	Data size for the MODBUS reading	-	Depends on the function used
Read Data Range	Range of addresses for the MODBUS data reading	-	0 to 2147483646
Write Data Start Address	Initial address for the MODBUS writing data	-	1 to 65536
Write Data Size	Data size for the MODBUS writing	-	Depends on the function used
Write Data Range	Range of addresses for the MODBUS data MODBUS writing	-	0 to 2147483647
Diagnostic Variable	Diagnostic variable name	-	Name of a variable declared on a program or GVL
Disabling Variable	Initial Variable used to disable	-	Field designed for the symbolic variable,

	the MODBUS relation		which is used to individually disable the configured MODBUS requests. This variable may be of BOOL type. The variable may be single or array element and can also be in structures.

Table 8-29 MODBUS Ethernet Client for Symbolic Mapping Relation Configuration

MODBUS Ethernet Client for Direct Representation (%Q)

By adding the MODBUS RTU Ethernet Server for Direct Representation protocol, the following screen will be shown.

Figure 8-67. MODBUS Ethernet Server for Direct Representation

Configuration	Description	Default	Possibilities
%Q Start Address of Diagnostic Area	Diagnosis variables initial address (%QB)	-	Any address of %Q area, limited according to CPU used
Size	Diagnosis variable size	20	20
TCP Port	TCP port	502	0 to 65535
Mapping Disabling	Initial address used to disable the MODBUS relations configured in the server Mappings field (%QX.X)	0	Any address of %Q area, limited according to CPU used
Protocol	Protocol selection	TCP	RTU via TCP TCP

Table 8-30. MODBUS Ethernet Server for Direct Representation Configuration

Advanced Configurations

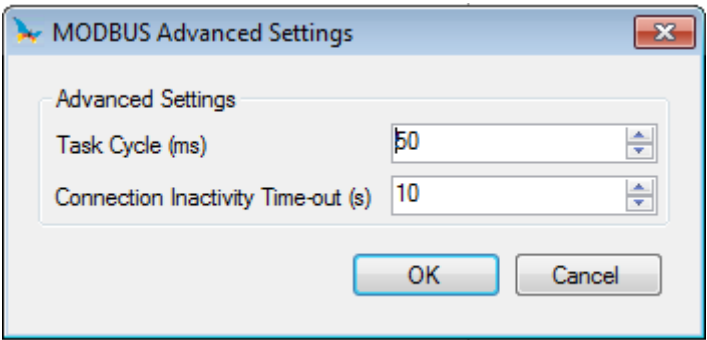


Figure 8-68. MODBUS Ethernet Server for Direct Representation Advanced Configurations Screen

Configuration	Description	Default	Possibilities
Task Cycle (ms)	Time for the instance execution within the cycle, without considering its own execution time	100	1 to 1000
Communication Timeout (ms)	Timeout TCP/IP	10	10 to 65535

Table 8-31. MODBUS Ethernet Server for Direct Representation Advanced Configurations

Adding a MODBUS Relation

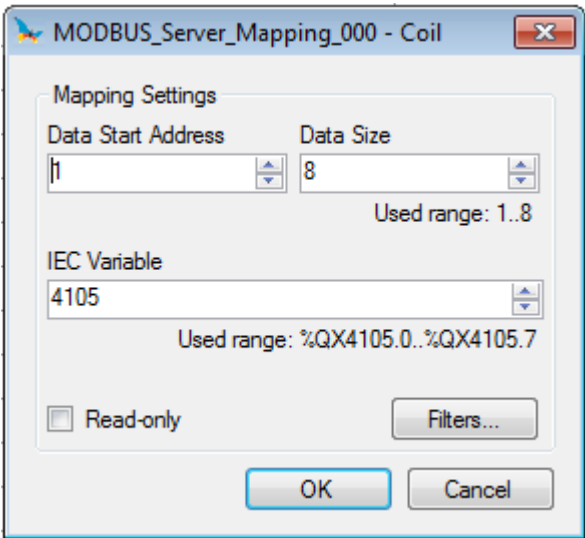


Figure 8-69. Adding a MODBUS Ethernet Server for Direct Representation Relation

Configuration	Description	Default	Possibilities
Data Type	MODBUS data type	Coil	Coil (1 bit) Holding Register (16 bits) Input Status (1 bit)

			Input Register (16 bits)
Data Start Address	MODBUS data initial address	0	1 to 65536
Data Size	MODBUS data quantity	8	1 to 65536 (Holding Register and Input Register) 8 to 65536 (Coil and Input Status)
IEC Variable	Variables initial address (%Q)	0	Any address area %Q, limited according to the CPU used

Table 8-32. MODBUS Ethernet Server for Direct Representation Relation Configuration

Filters

Figure 8-70. Filters

Configuration	Description	Default	Possibilities
Write Filter IP	Specifies a IP interval with writing access to the declared variables in the MODBUS relation	0.0.0.0	0.0.0.1 to 255.255.255.254
Write Filter Mask	Specifies a subnet mask and the parameter Write Filter IP	0.0.0.0	0.0.0.1 to 255.255.255.254
Read Filter IP	Specifies a IP interval with reading access to the declared variables in the MODBUS relation	0.0.0.0	0.0.0.1 to 255.255.255.254
Read Filter Mask	Specifies a subnet mask and the parameter Read Filter IP	0.0.0.0	0.0.0.1 to 255.255.255.254

Table 8-33. Filters Configuration

NOTE: Online changes may not be applied when you change MODBUS mappings parameters or when you add or remove devices, mappings or MODBUS instances.

MODBUS Ethernet Server for Symbolic Mapping

By adding a MODBUS Ethernet Server for Symbolic Mapping device, its configuration screen is showed.

General Parameters

The general parameters found on the initial configuration screen of the MODBUS Ethernet Server for Symbolic Mapping protocol (Figure 8-71), are defined as:

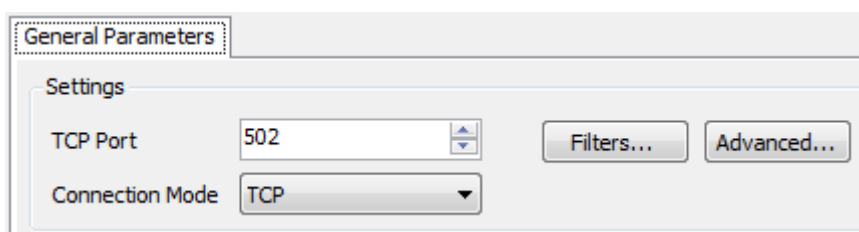


Figure 8-71 MODBUS Ethernet Server for Symbolic Mapping General Parameters Screen

Configuration	Description	Default	Possibilities
TCP Port	TCP Port	502	2 to 65534
Connection Mode	Protocol selection	TCP	RTU via TCP TCP

Table 8-34 MODBUS Ethernet Server for Symbolic Mapping General Configuration

The configurations in the *Filters...button*, described on Table 8-33, are related to the TCP communication filters.

The MODBUS Ethernet Server for Symbolic Mapping protocol communication times, found in the *Advanced... button* on the configuration screen, are divided in: Task Cycle and Connection Inactivity Timeout.

Configuration	Description	Default	Possibilities
Task Cycle (ms)	Time for the instance execution within the cycle, without considering its own execution time	50	5 to 100
Connection Inactivity Timeout (s)	Maximum idle time between client and server before the connection being closed by the server.	10	10 a 3600

Table 8-35. MODBUS Ethernet Server for Symbolic Mapping Advanced Configurations

Mapping Configuration

The mapping configuration on the MODBUS Ethernet Server for Symbolic Mapping, seen on Figure 8-72, follow the parameters described on Table 8-36:

	Value Variable	Data Type	Data Start Address	Data Size	Data Range
*					

Figure 8-72. MODBUS Ethernet Server for Symbolic Mapping Screen

Configuration	Description	Default	Possibilities
Variable Value	Name of the symbolic variable	-	Name of a variable declared in a program or GVL
Data Type	MODBUS data type	-	Coil Input Status Holding Register Input Register
Data Start Address	MODBUS data initial address	-	1 to 65536
Data Size	MODBUS data size	-	1 to 65536
Data Range	Range of addresses of the configured data	-	-

Table 8-36 MODBUS Ethernet Server for Symbolic Mapping Configuration

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.

CPU Editor

The CPU-related parameters are configured as the screen below. The screen is in the device tree.

General Parameters

Diagnostics Area
 %Q Start Address: 21520
 Size: 520
 Used range: %QB21520..%QB22039

Retain Area
 %Q Start Address: 4096
 Size: 8192
 Used range: %QB4096..%QB12287

Persistent Area
 %Q Start Address: 12288
 Size: 8192
 Used range: %QB12288..%QB20479

CPU Parameters
 Start User Application after a Watchdog Reset: Enabled
 Hot Swap Mode: Enabled, without startup consistency

TCP/IP Parameters
 Initial Time-out (x100 ms): 6
 ACK Delay (x10 ms): 19

Project Parameters
☐ Consist retain and persistent area in %Q.
 Memory Card...

Figure 8-73. 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.

Serial Interfaces

The COM1 and COM2 serial interfaces are configured in the following screen. They are located in the devices tree, below the CPU.

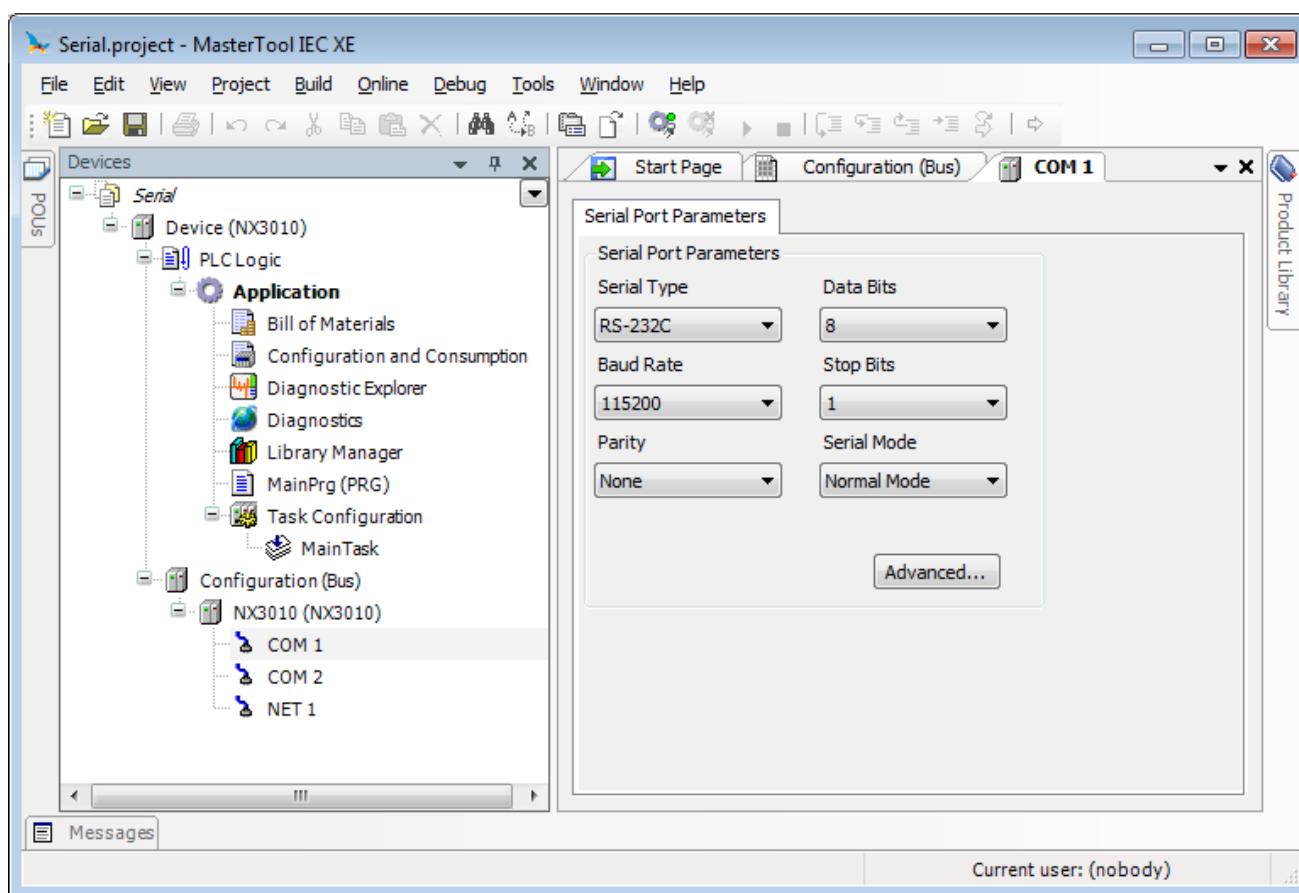


Figure 8-74. Serial Interfaces

For information on parameters, features and configuration possibilities see the current CPU manual.

Ethernet Interfaces

The Ethernet interfaces are configured in the following screen. They are located in the devices tree, below the CPU.

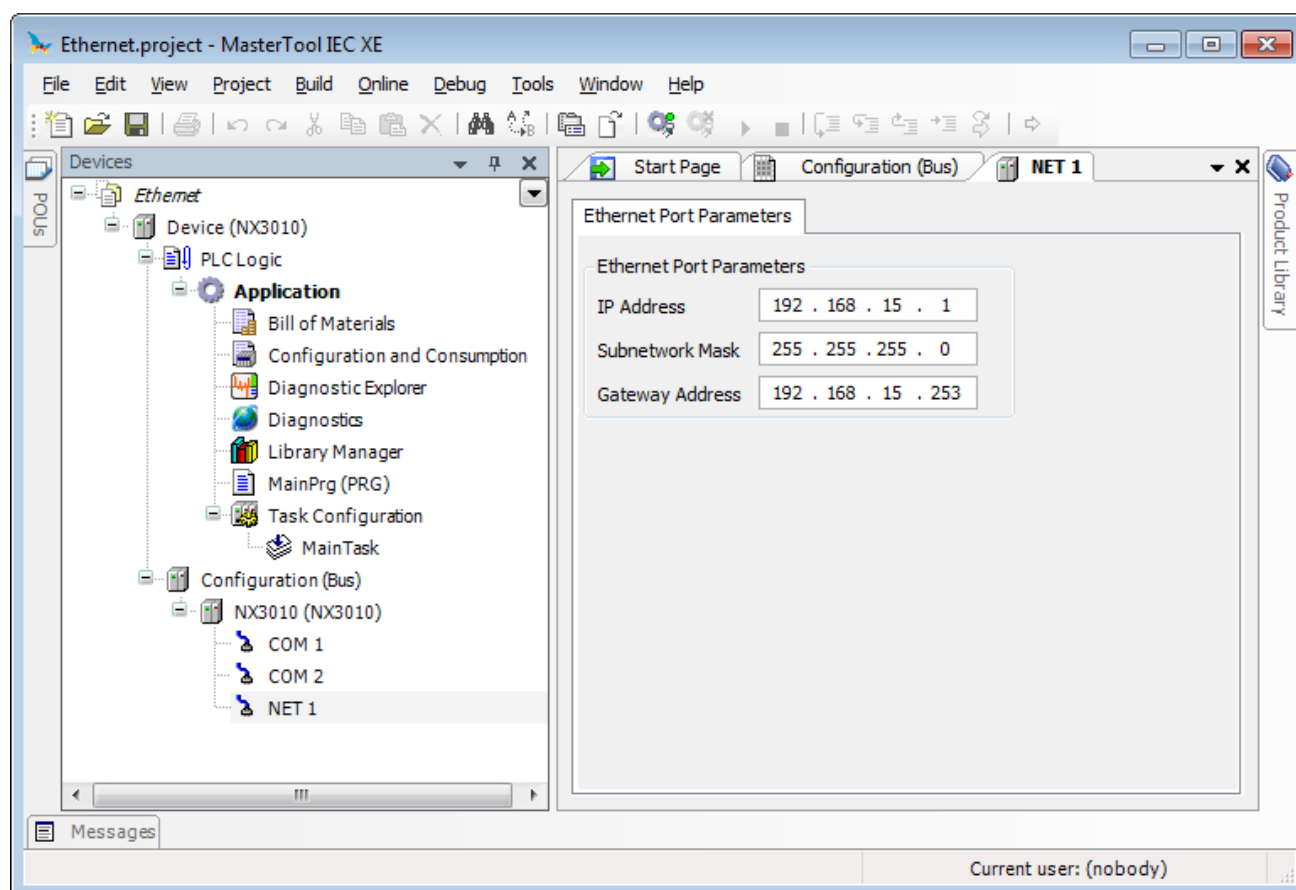


Figure 8-75. Ethernet Interface

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.

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 (Figure 8-76).

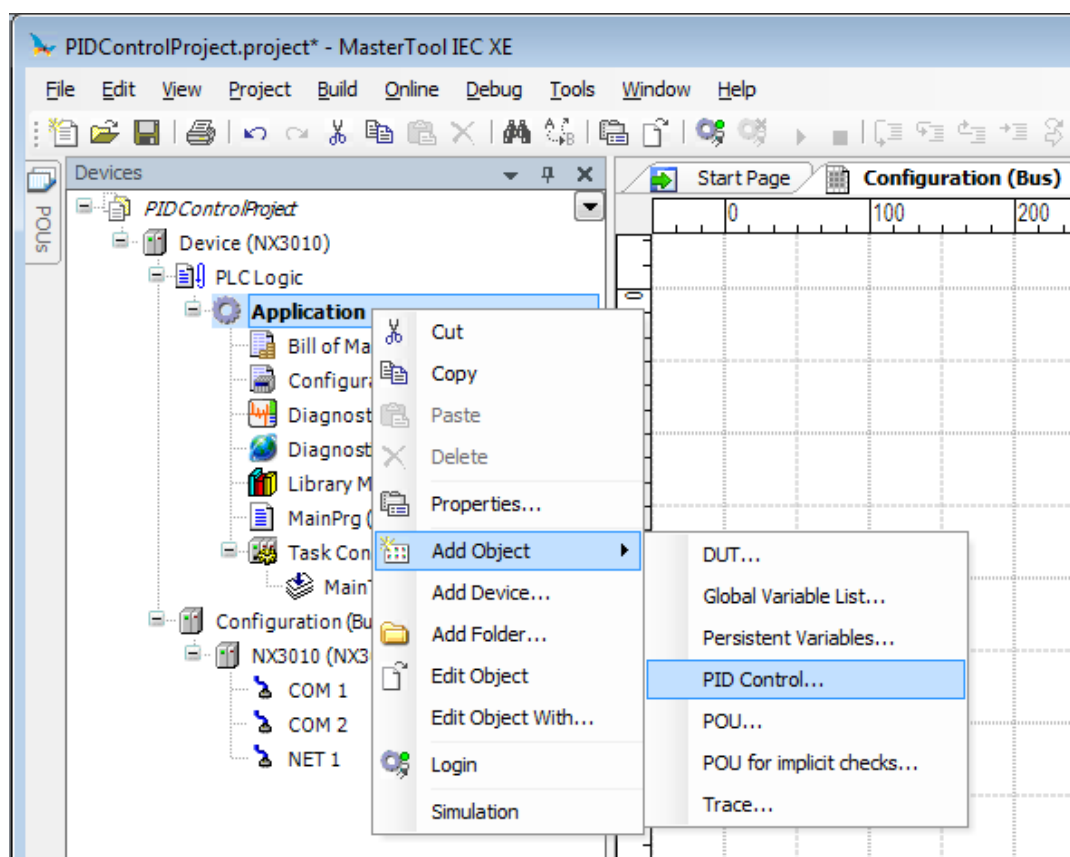


Figure 8-76. Steps to inserting the PID Control object in an application

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.

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 8-77 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.

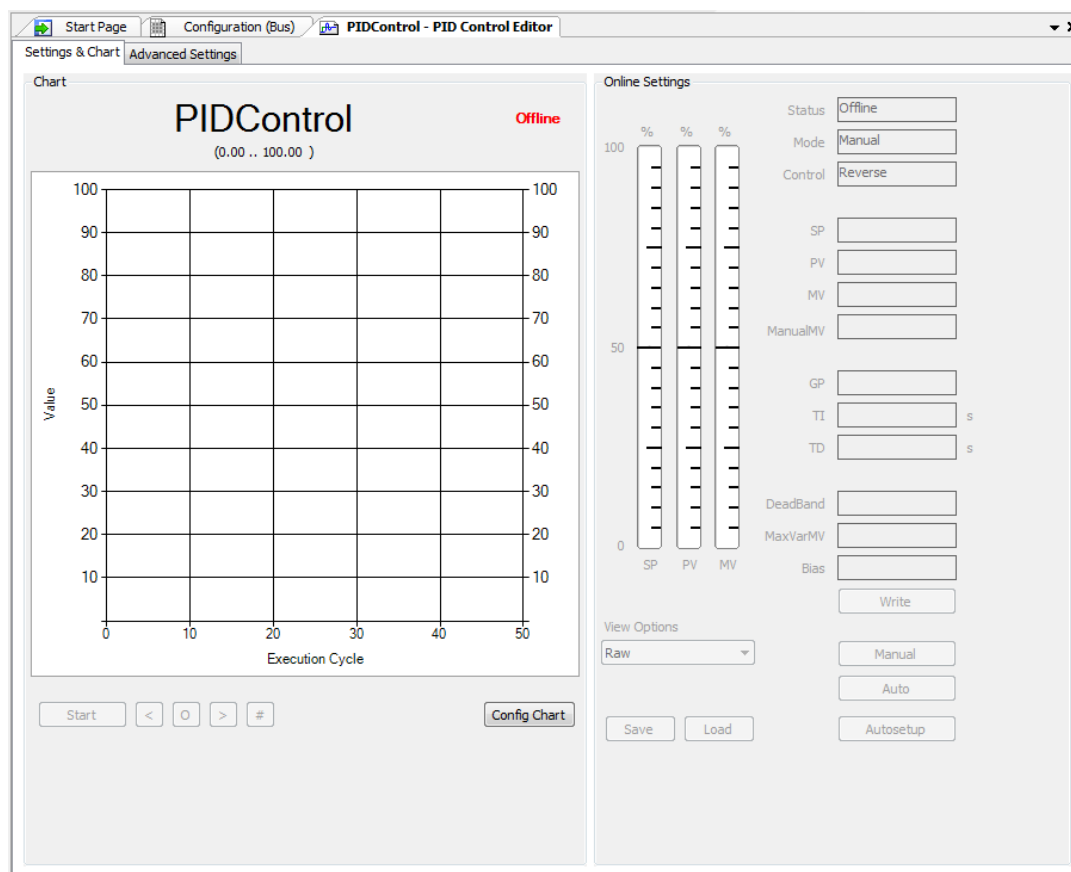


Figure 8-77. Graphical environment of the PID Control object

The graphical environment illustrated by Figure 8-77 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 Figure 8-78.

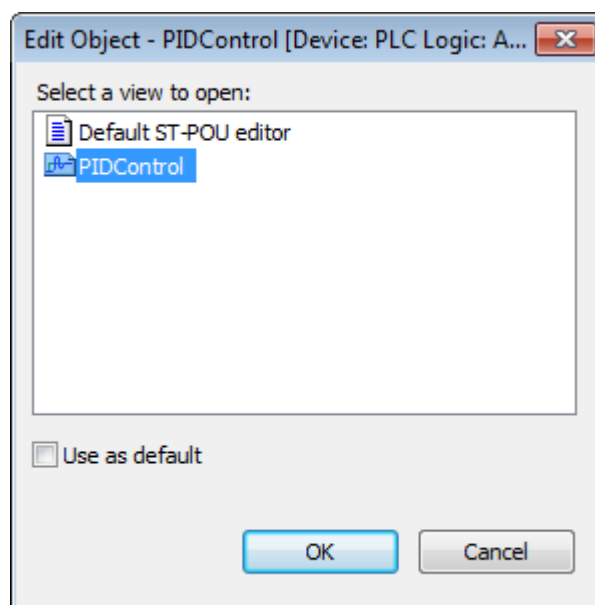


Figure 8-78. Access to the graphical environment of PID Control object

Tab Settings & Chart

The Figure 8-77 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*.

Group: Chart

This group is responsible for displaying the process trend chart, including some operations and/or settings in this graph. The Figure 8-79 presents this group.

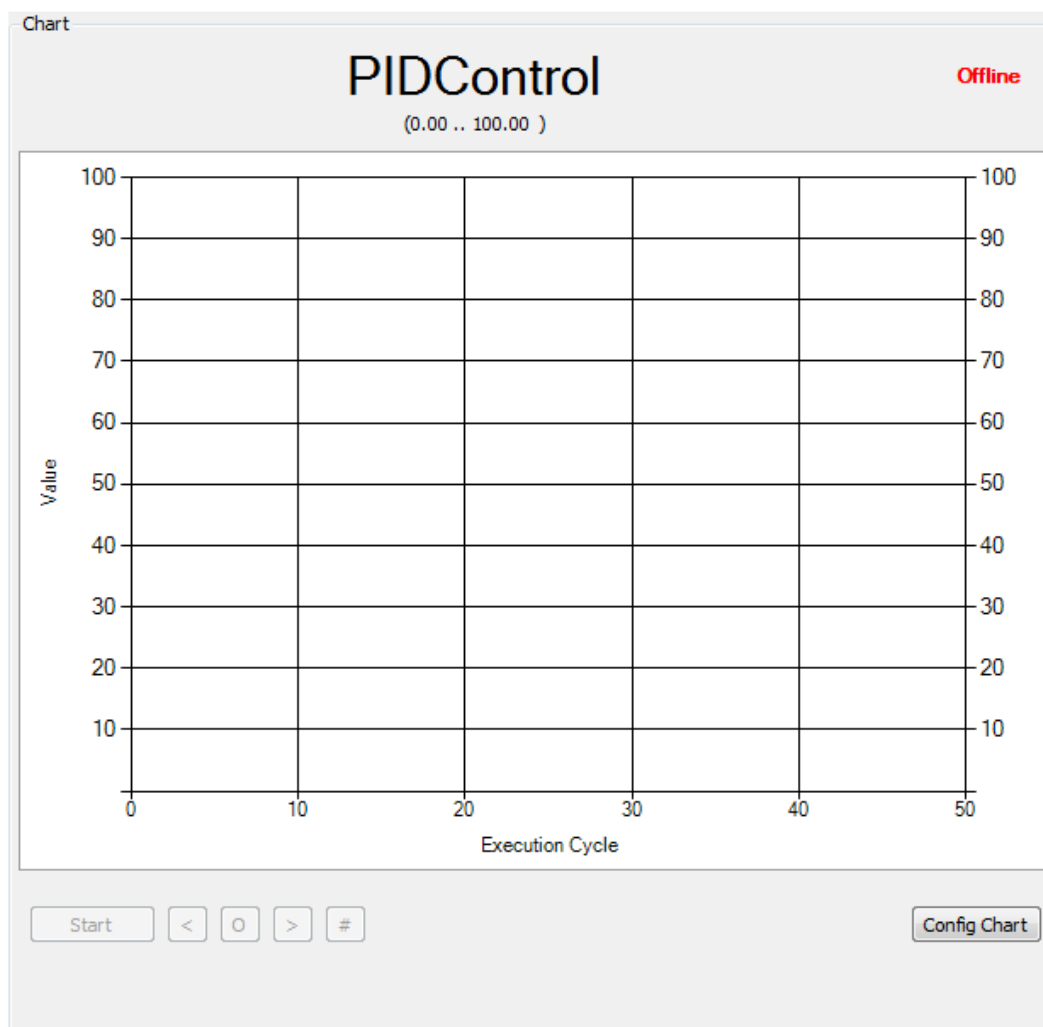


Figure 8-79. Group Graphic

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 8-80 shows these buttons.



Figure 8-80. 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 8-81.

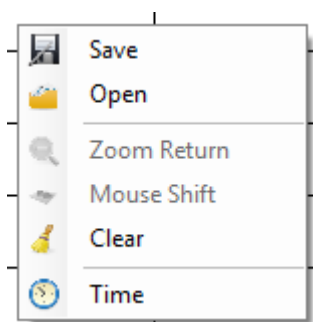


Figure 8-81. The Context Menu of the trend chart

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.

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 8-82 shows this window.

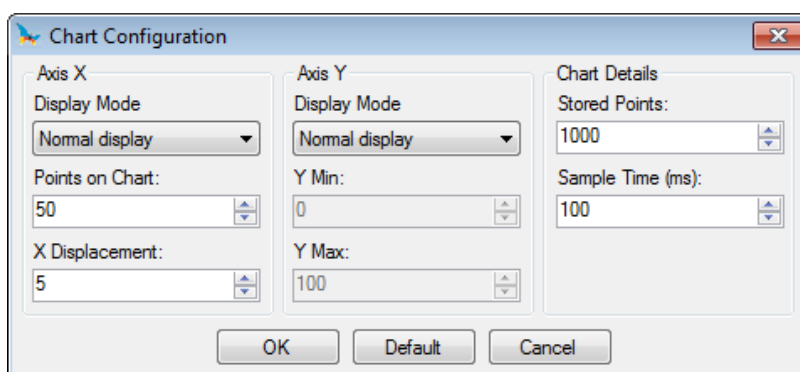


Figure 8-82. Chart Configuration Window

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 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.

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 8-83 displays details of the *Online Settings* group.

Online Settings

100 % % %

50

0 SP PV MV

Status

Mode

Control

SP

PV

MV

ManualMV

GP

TI s

TD s

DeadBand

MaxVarMV

Bias

Write

Manual

Auto

Autotsetup

View Options

Save Load

Figure 8-83. Online Settings group

In the upper left corner of the Figure 8-83, 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 (Figure 8-84) 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.

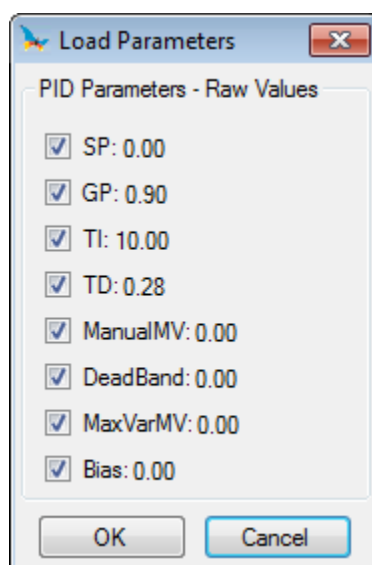


Figure 8-84. Parameters Selection Window

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.

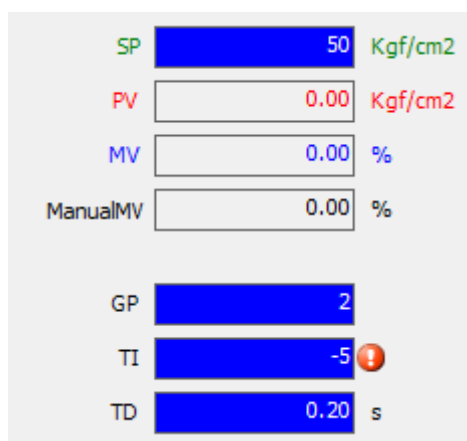


Figure 8-85. Parameters Editing Example

The Figure 8-85 demonstrates that the fields *SP*, *GP*, *TI* and *TD* have changed, but have not yet had their new values sent to the PLC. Notice that the *TI* 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 Figure 8-86 displays the preview of these fields after the *Write* button is pressed.

SP	50.00	Kgf/cm2
PV	0.00	Kgf/cm2
MV	0.00	%
ManualMV	0.00	%
GP	2.00	
TI	-5	
TD	0.20	s

Figure 8-86. 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 Figure 8-87.

Figure 8-87. Restoring the Value of a Field

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 8-88 displays this tab.

Input/Output Settings		Variable	Minimum	Maximum	Unit
SP Raw			0.00	30000.00	
SP Engineering			0.00	100.00	
PV Raw	PV_VARIABLE_NAME		0.00	30000.00	
PV Engineering			0.00	100.00	
MV Raw	MV_VARIABLE_NAME		0.00	30000.00	

Control Settings	
Sample Time (ms):	Control
100	Reverse
☒ Enable proportional action ☒ Enable integral action ☒ Enable derivative action ☒ Enabled derivative action on PV	

Project Settings	
Autosetup Restrictions	
☒ Automatic Task Association	

Figure 8-88. Advanced Settings tab

Observing the Figure 8-88 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.

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 8-89 shows the group.

	Variable	Minimum	Maximum	Unit
SP Raw		0.00	30000.00	
SP Engineering		0.00	100.00	
PV Raw	PV_VARIABLE_NAME	0.00	30000.00	
PV Engineering		0.00	100.00	
MV Raw	MV_VARIABLE_NAME	0.00	30000.00	

Figure 8-89. 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*.

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.

Control Settings

Sample Time (ms): 100

Control: Reverse

☒ Enable proportional action

☒ Enable integral action

☒ Enable derivative action

☒ Enabled derivative action on PV

Figure 8-90. Group: Control Settings

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.

Project Settings Autosetup Restrictions

☒ Automatic Task Association

Figure 8-91. Tab: Project Settings

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.

Project Settings Autosetup Restrictions

Minimum MV 15 %

Maximum MV 85 %

Minimum PV 10 %

Maximum PV 90 %

Figure 8-92. Tab: Project Settings

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 Figure 8-93 displays the window *Autosetup*.

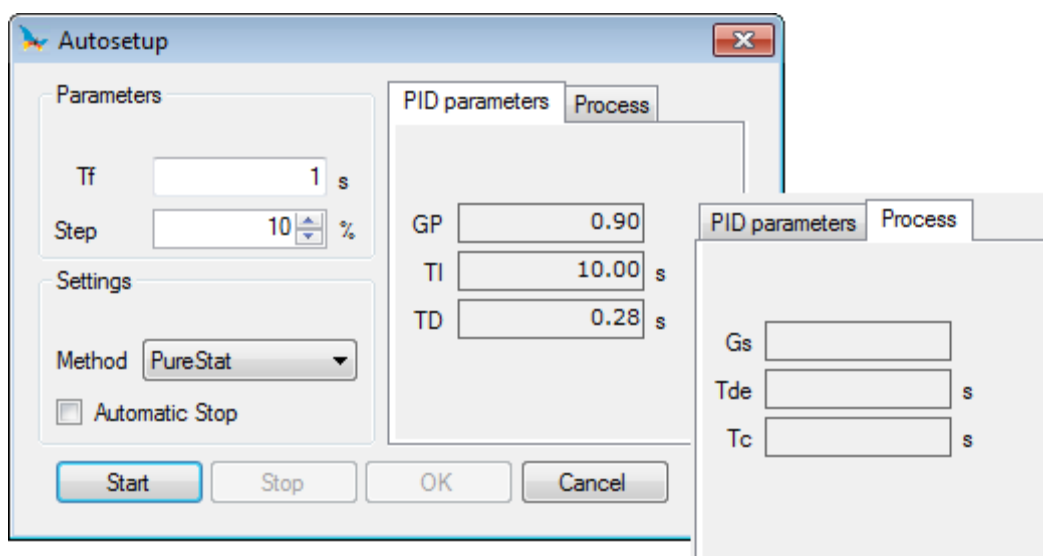


Figure 8-93. Window Autosetup

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 $T_f < T_c < 10$ and when $T_c / (T_{de} + \text{SampleTime}/2) < 10$. Where T_c and T_{de} 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 $(MV + MV \cdot \text{Step})$.

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 *G_s*, *T_c* and *T_{de}* 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 sent 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 execute the MasterTool IEC XE development software installation, it is necessary to have the distribution CD-ROM or download the installation file from the site <http://www.altus.com.br>. Then close all programs that might have been executed in your PC and double-click the installation file. The following installation screen will appear



Figure 9-1. Installation Screen

Select the installation language and press OK to continue with the process. This action will trigger the MasterTool IEC XE installation. Then, the following screens will appear:



Figure 9-2. Initial Screen

The license contract screen will appear. It must be read carefully. In case of agreement with the license terms, click the option that accepts it and press *Next* to continue.

On the next screen it is possible to define whether or not a shortcut on the Desktop will be created. Select the desired option then press *Next* to continue.

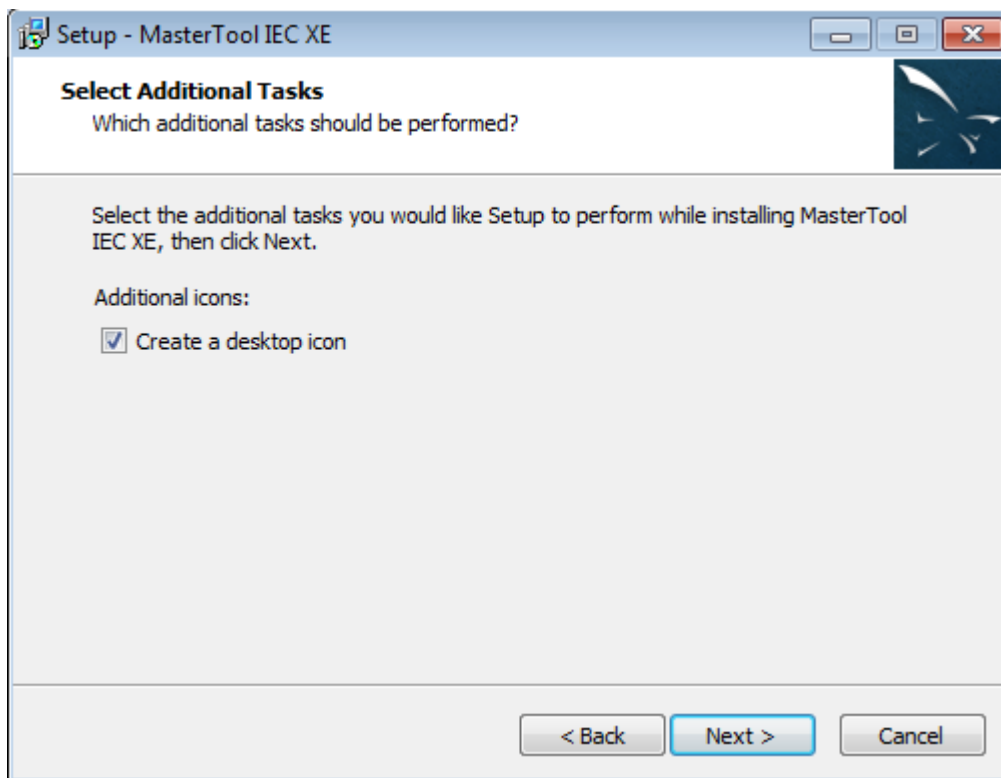


Figure 9-5. Icon on the Desktop Selection Screen

On the next screen a review of the selected components to be installed is executed. Click in *Next* to continue or *Back* to modify any feature.

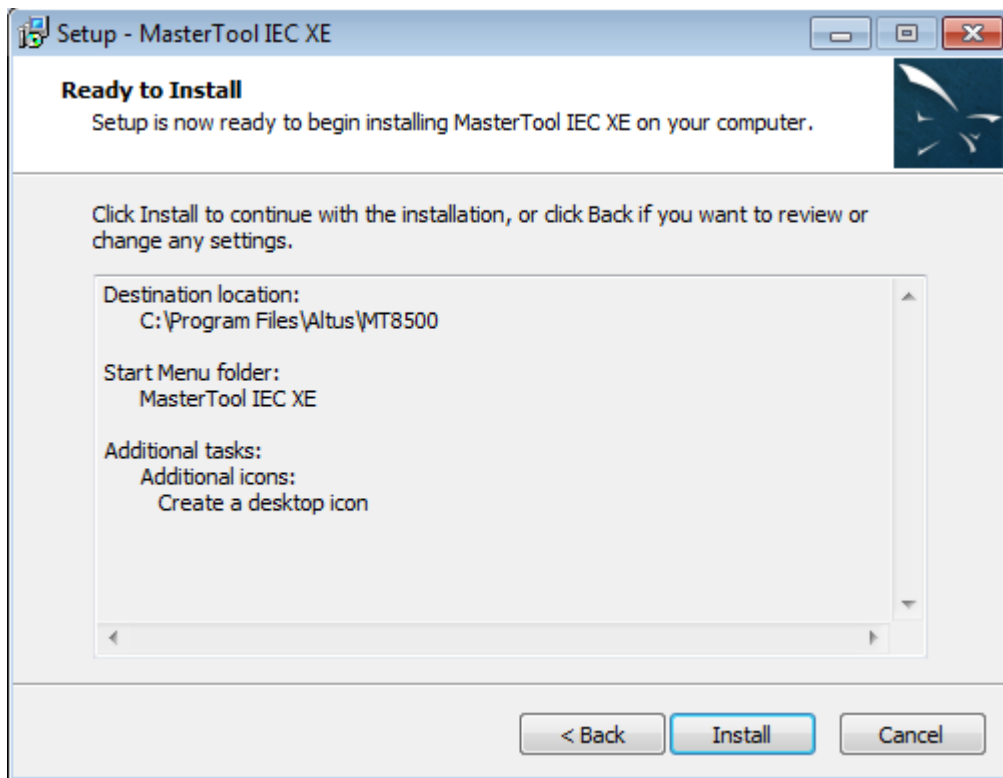


Figure 9-6. Components Revision Screen

At first the software prerequisites, such as the .NET Framework have to be installed.

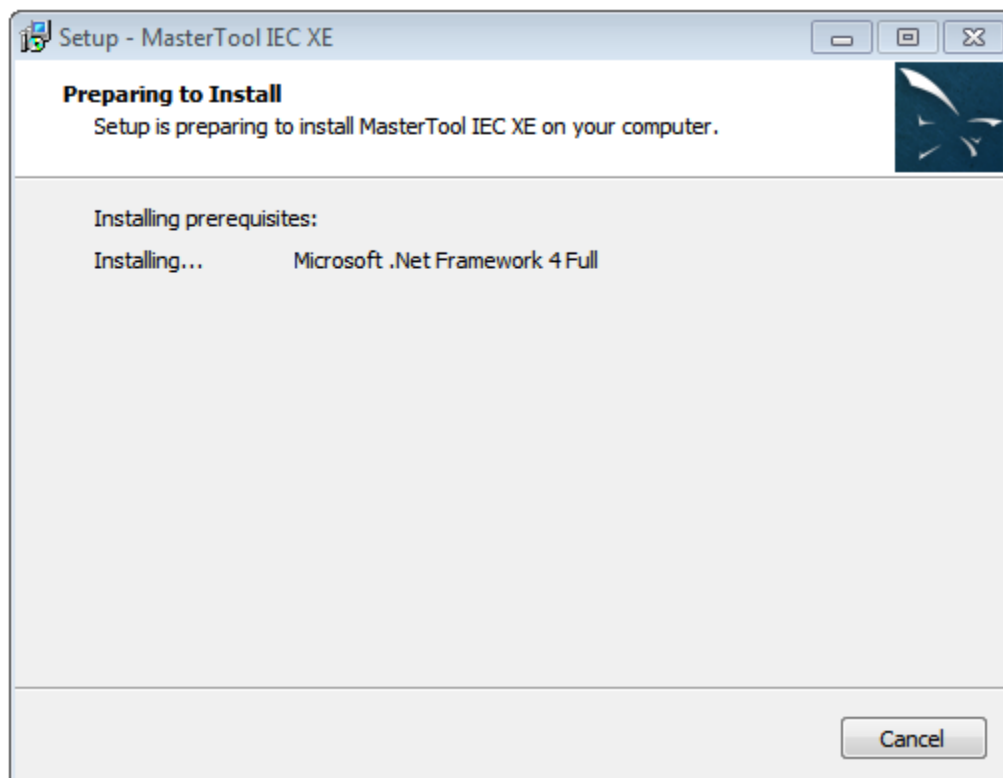


Figure 9-7. Prior Conditions Screen to Install MasterTool IEC XE

During the installation process, a screen requesting the extraction of the .NET Framework files will appear.

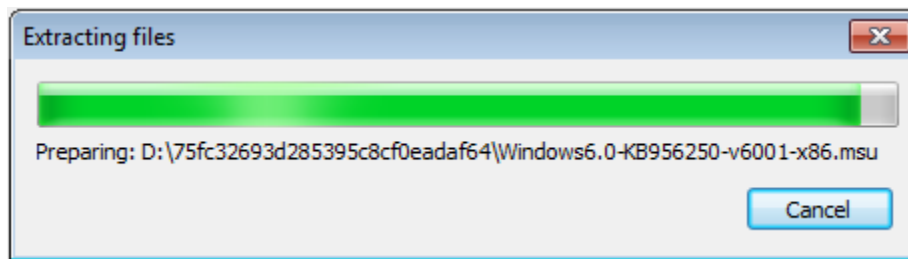


Figure 9-8. Extracting Files for the .NET Framework Installation

The Microsoft .NET Framework installation will be started.

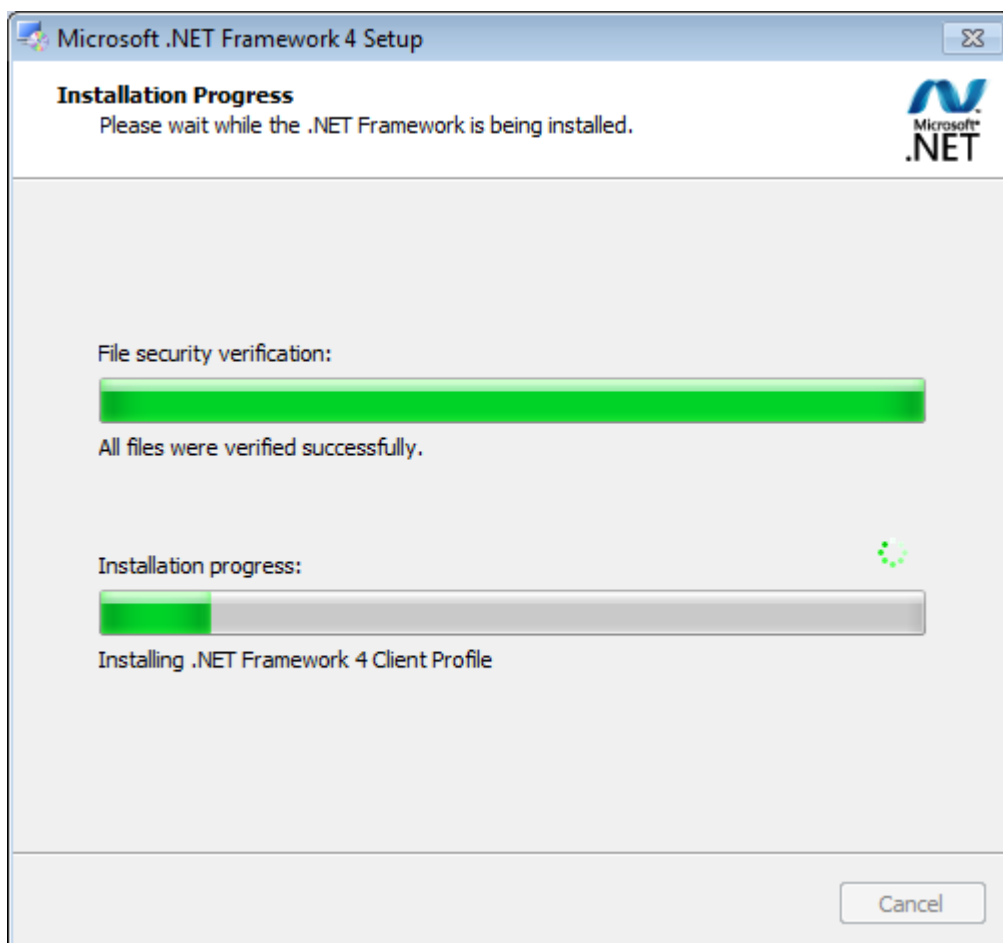


Figure 9-9. .NET Framework Installation Screen

At this point, MasterTool IEC XE installation was started and the required files are being installed on the computer. This operation may take some minutes, dependind on its configuration.

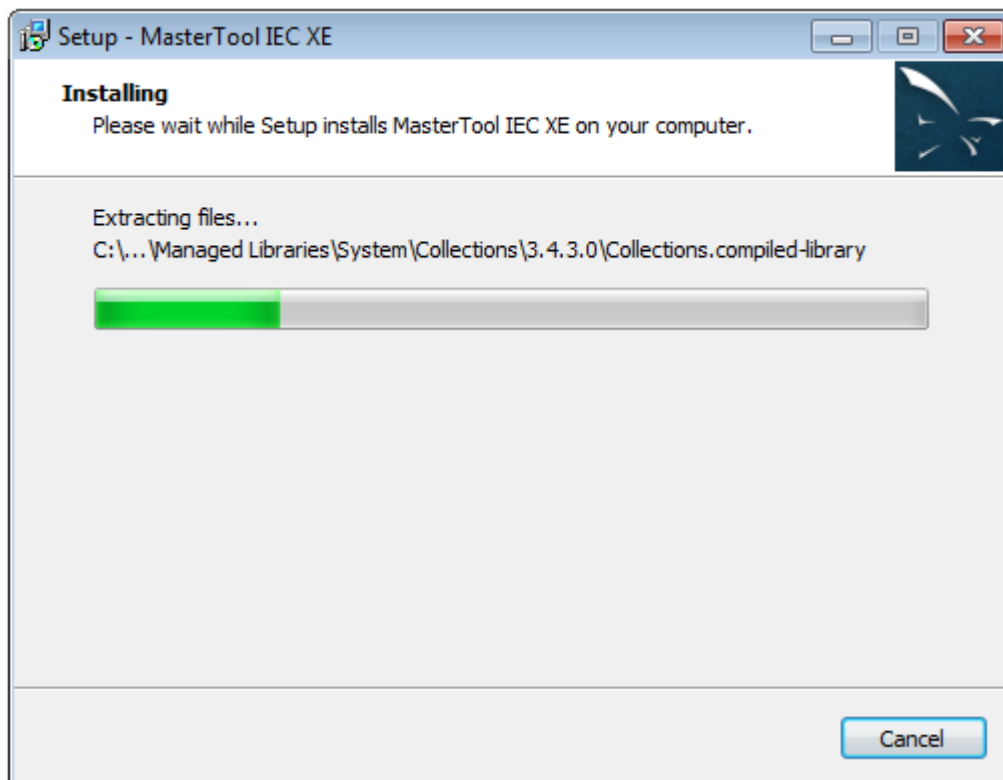


Figure 9-10. MasterTool IEC XE Installation Screen

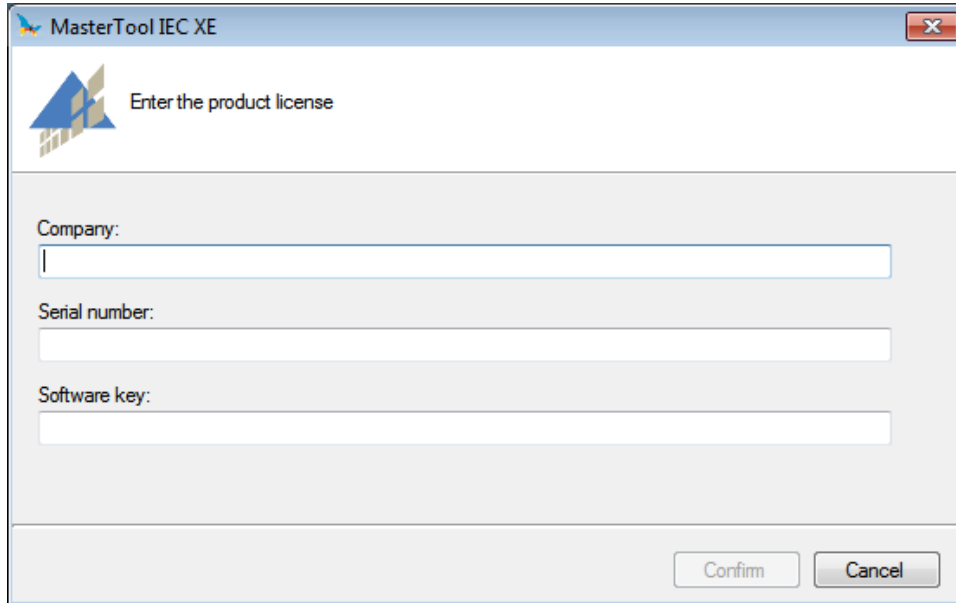
After installation is finished the next screen will be shown. On this window it is strongly recommendable to choose the restart option. After that, click *Finish* to conclude the installation procedure.



Figure 9-11. Installation Complete

The MasterTool IEC XE will be installed and ready to use. To execute, click on the “MasterTool IEC XE” shortcut created during the installation, in *Start Menu*.

The first time the software is initialized, a screen requesting registry information will appear. After filling the fields correctly, click on *Confirm* to use the MasterTool IEC XE.

A screenshot of the MasterTool IEC XE application window. The window has a title bar with the text "MasterTool IEC XE" and a close button. Inside the window, there is a logo on the left and the text "Enter the product license" on the right. Below this, there are three input fields labeled "Company:", "Serial number:", and "Software key:". At the bottom right of the window, there are two buttons: "Confirm" and "Cancel".**Figure 9-12. Registry Information Screen**

In case a version of MasterTool IEC XE is already set on your computer the installer will ask about the uninstallation of this version before the new one gets installed. If the installed version is lower than version 1.40 the uninstallation process is different and after its completion it will be necessary to reinstall it.

10. Diagnostics

Diagnostics Page

The Diagnostics page, or Diagnostics Explorer, refers to the diagnosis the inclusion via WEB within MasterTool IEC XE, so that the access is faster and more objective.

The access to the characteristic occurs in two ways:

- Access the *Diagnostics Explorer* option in the device tree (located in the devices Tree). The diagnostics web page will be automatically loads when you open the screen. If not, type the correct IP in the field indicated on the following picture. Remember that in order to the diagnostics page to be shown, the user must have a defined CPU as active path (For further information see **Communication Settings**).

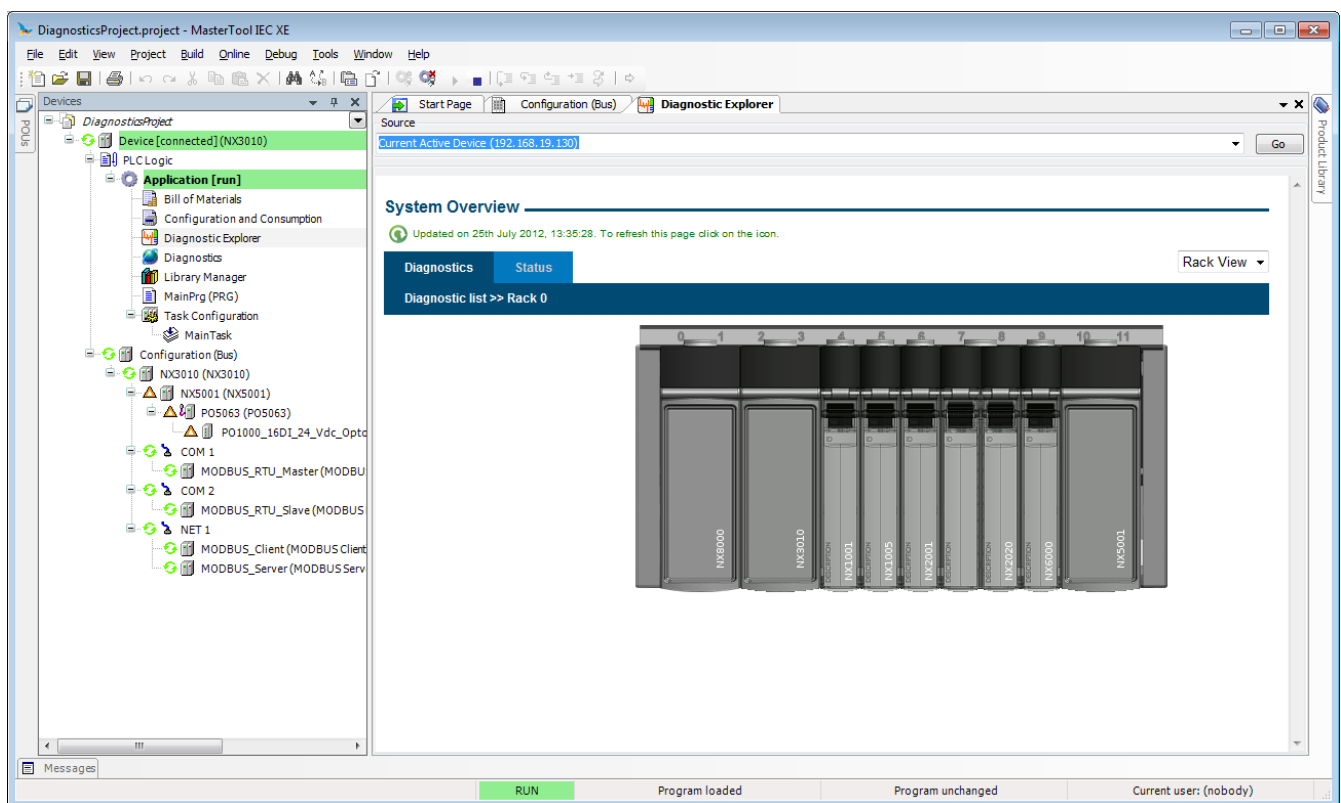


Figure 10-1. Diagnostics Explorer

- Click the right mouse button on the module and select *Diagnostics*. The *Diagnostics Explorer* will be opened and will take you to the status page of the respective module.

Diagnostics

The *Diagnostics* object (located in the devices tree) is the where the Diagnostics Global Variables of all currently used devices are shown.

Each used device, has diagnostics variables and they are exhibited as shown in the following screen.

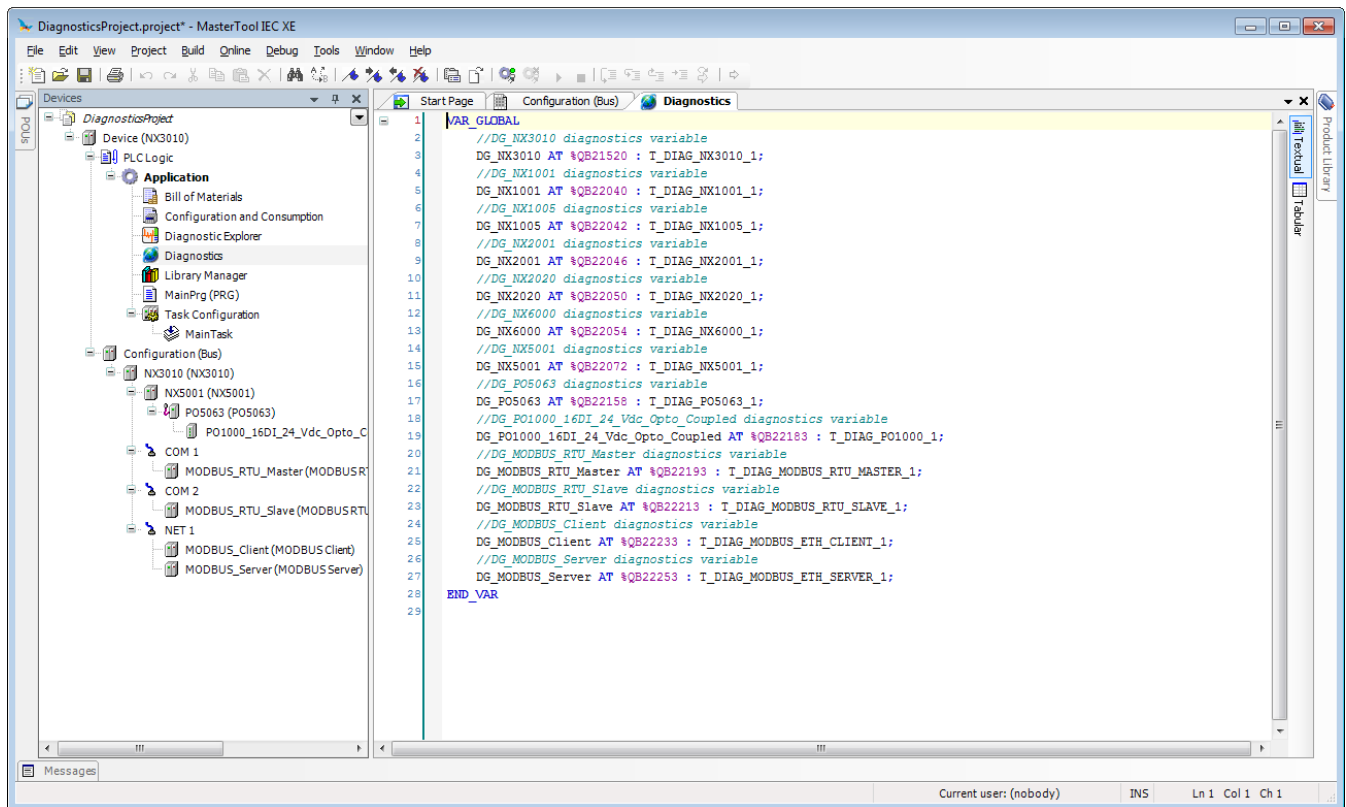


Figure 10-2. Diagnostics Variables

When a device is removed its diagnostics variables are also automatically removed.

Two other objects are created in the devices tree for Symbolic Mapping communication drivers. They are named *Disables* and *ReqDiagnostics*.

In addition to that, the diagnostic variables of these drivers are declared on the *Diagnostics* object and do not use the AT directive in its declaration because there is no relation between a symbolic variable and a direct representation variable for these devices. This behavior can be observed on Figure 10 3.

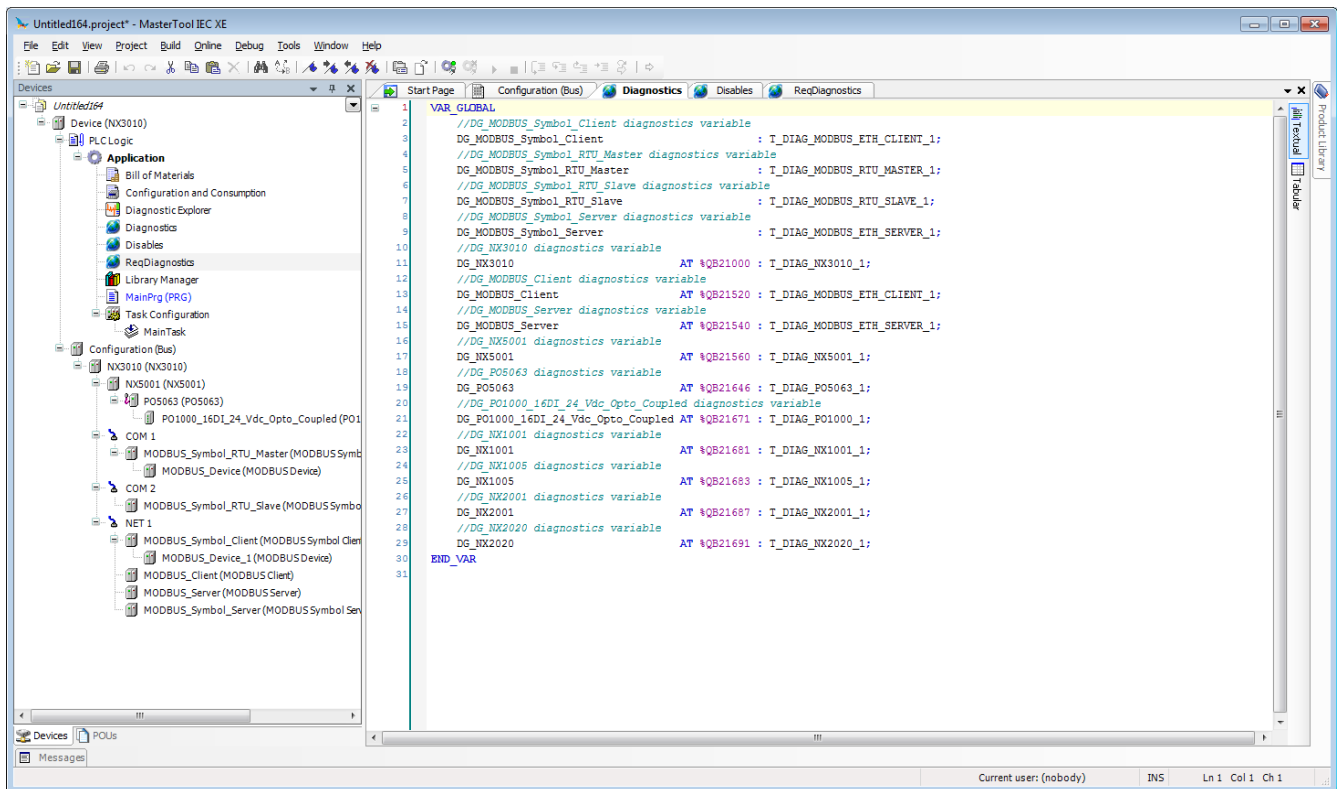


Figure 10-3. Diagnostics Variables Using Communication Drivers Through Symbolic Mapping

The *Disables* object is used to declare disabling requests variables in drivers, which use symbolic mapping. These variables can be declared directly by the user in any POU or GVL, but can also be generated automatically using the *Generate Disabling Variables* button available on the screen of the drivers request configuration. If the object does not exist in the project yet it is created after pressing the button. After that, if necessary the variables declared in this object may be edited.

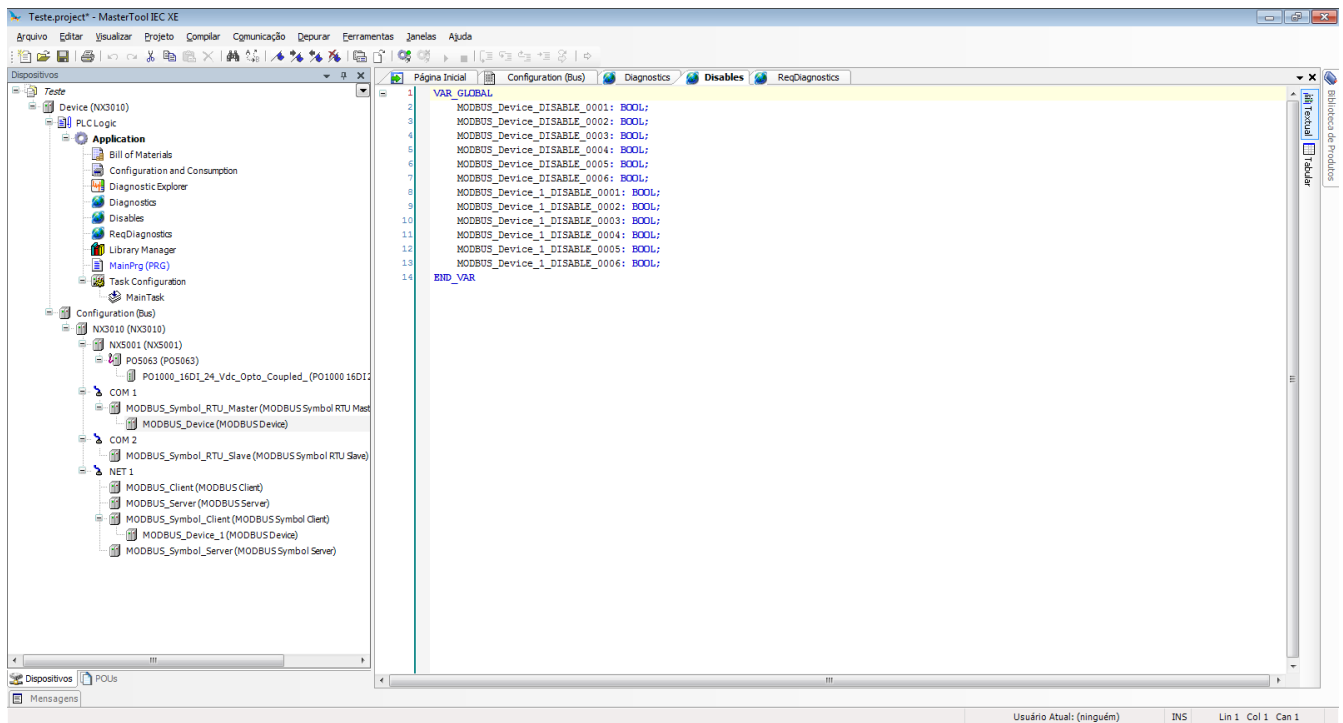


Figure 10-4. Disables Object

The *ReqDiagnostics* object is used to declare request diagnostic variables. These variables can be declared directly by the user in any POU or GVL, but can also be generated automatically using the *Generate Diagnostic Variables* button available on the screen of the drivers request configuration. If the object does not exist in the project yet it will be created after the button is pressed. After that, if necessary, the variables declared in this object may be edited.

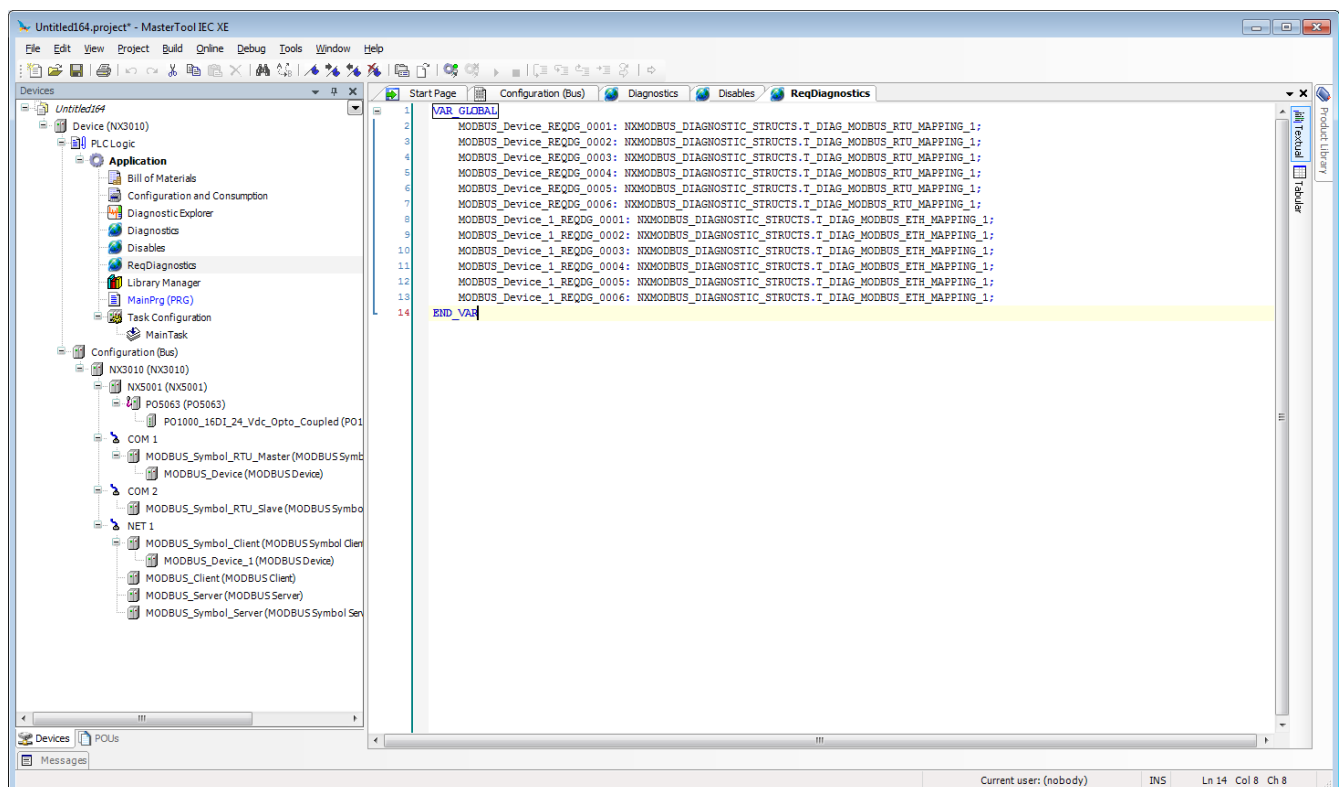


Figure 10-5. ReqDiagnostics Object

11. Glossary

Active CPU	In a redundant system, the active CPU performs the control of the system by reading the values of points of entry, running the program application and triggering the values of outputs.
Algorithm	Finite sequence of well-defined instructions to solve problems.
Application Program	Is the program loaded into a PLC, which determines the operation of a machine or process.
Backup	Data backup.
Bit	Basic information unit, it may be at 1 or 0 logic level.
Breakpoint	Breakpoint in application for debugging.
Byte	Information unit composed by eight bits.
CPU	Central Processing Unit. It controls the data flow, interprets and executes the program instructions as well as monitors the system devices.
Bus	Set of interconnected I/O modules to a CPU or to a head of network field.
Communication network	Set of equipment (nodes) interconnected by communication channels.
Context Menu	Dynamic Menu with content according to the current context.
Cycled	PLC execution mode step-by-step, where each step is a cycle.
Default	Preset value for a variable, used in case there is no definition.
Diagnostics	Procedure used to detect and isolate faults. It is also the set of data used for this determination, which serves for the analysis and remediation.
Download	Load configuration or program in the PLC.
Frame	A unit of information transmitted on the network.
Gateway	Equipment for connecting two communication networks with different protocols.
Hardware	Physical equipment used in data processing which normally run programs (software).
Hyperlink	Shortcut navigation for a new help page.
IEC 61131	Generic standard for operation and use of PLCs. Former IEC1131.
Interface	Adapts electric and/or logically the transfer of signals between two devices.
Interruption	Priority event that temporarily halts the normal execution of a program. The interruptions are divided into two generic types: hardware and software. The former is caused by a signal coming from a peripheral, while the later is caused within a program.
Input/output	Also called I/O. I/O devices on a system. In the case of PLCs typically correspond to modules digital or analogue output or input monitoring or driving the controlled device.
I/O	See Input/Output.
I/O Module	Module belonging to subsystem of inputs and outputs.
Local bus	Set of interconnected I/O modules to a CPU.
Local host	Machine, PC or the system that is in use.
Login	Action to establish a communication channel with the PLC.
Master	Equipment connected to a communications network where originate command requests to other network equipment.
Multicast	Simultaneous communication with a group of nodes connected to a network.
kbytes	Memory size unit. Represents 1024 bytes.
LED	Light Emitting Diode. Type of semiconductor diode that emits light when energized. It's used for visual feedback.
Menu	Set of options available and displayed by a program in the video and that can be selected by the user to activate or to perform a particular task.
Module (hardware)	Basic element of a system with very specific functionality. It's normally connected to the system by connectors and may be easily replaced.
Node	Any station of a network with communication skills using an established protocol.
Operands	Elements on which the instructions work. Can represent constants, variables, or a set of variables.
Ping	Packet Internet Network Grouper, is a command used by the ICMP protocol used to test connectivity between devices and was created for use in networks with protocol stack TCP / IP.
PDO	Process Data Object, is a message of CAN protocol that contains the operational data.
PLC	Acronym for programmable controller, See Programmable Controller
POU	Program Organization Unit, is a subdivision of the application program that can be written in any of the available languages.
Programmable Controller	Also known as PLC. Equipment controlling a system under the command of an application program. It is composed of a CPU, a power supply and I/O modules.

Programming Language	A set of rules and conventions used for the elaboration of a program.
Protocol	Procedural rules and conventional formats which, upon control signals, allow the establishment of a data transmission and error recovery between equipment.
RAM	Random access memory. Is the memory where all addresses can be accessed directly in a random way and with the same speed. Is volatile, i.e., its content is lost when the equipment is de-energized, unless if you have a battery for retaining values.
Remote Bus	Set of I/O Modules connected to the Fieldbus head.
Reset	Command to reboot the PLC.
RUN	Command to put PLC in execution mode.
Scan Cycle	A complete execution of the PLC application program.
Serial Channel	Unit interface that transfers data serially.
Set	Action to assign the logical high level state to a boolean variable.
Slave	Equipment connected to a communications network that transmits data only if it is requested by other equipment called master.
Software	Computer programs, procedures and rules related to the operation of a data processing system.
STOP	Command to freeze the PLC in its current state.
Sub network	Segment of a communication network that connects a group of devices (nodes) with the goal of isolating the local data traffic or using different protocols or physical media.
Timeout	Maximum preset time to a communication to take place. When exceeded, then retry procedures are started or diagnostics are activated.
Tooltip	Text box with a help or where you can enter with the help.
Tree	Data structure for hardware configuration.
Upload	Reading of the program or configuration from the PLC.
Visualization	Set of screens of PLC.
Watchdog	Electronic circuit that checks the equipment operation integrity.
XML	Extensible Markup Language: is a standard for generating markup languages.
Zoom	In the context of keyboard function window, is used for the exchange of screens.