



MasterTool IEC XE User Manual MT8500

Rev. F 12/2018
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.

The examples and figures presented in this document are solely for illustrative purposes. Due to possible upgrades and improvements that the products may present, Altus assumes no responsibility for the use of these examples and figures in real applications. They should only be used to assist user trainings and improve experience with the products and their features.

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

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

Altus does not acknowledge any other guarantee, directly or implied, mainly when end customers are dealing with third-party suppliers.

The requests for additional information about the supply, equipment features and/or any other Altus services must be made in writing form. Altus is not responsible for supplying information about its equipment without formal request.

COPYRIGHTS

Nexto, MasterTool, Grano and WebPLC are the registered trademarks of Altus Sistemas de Automação S.A.

Windows is registered trademark of Microsoft Corporation.

Summary

SUMMARY	II
1. INTRODUCTION	9
Documents Related to this Manual	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 and Recommended Requirements	14
Data for Purchase	15
Included Items	15
Product Code.....	15
3. CONCEPTS AND BASIC COMPONENTS	16
Introduction	16
Basic Concepts	16
Advanced Functionalities	16
Object Orientation in Programming and Project Structure	17
Special Data Types.....	17
Operators and Special Variables	17
New User Management and Access Rights Concept	17
Features in Editors.....	17
Library Versions.....	17
Additional Functionalities	17
Project Profiles	18
Project	18
Project Template	18
POUs Window	19
Project Settings and Project Information	19
External File	19
Device, Device Tree	20
Generals.....	20
Installing of Devices on the Local System	22
Arranging and Configuring Objects in the Devices Tree - Rules	22
Application	23
Task Configuration	23
Important Notes for Multitasking Systems	23
Code Generation and Online Change	24
Code Generation and Compile Information	24
Online Change.....	24
Boot Application (Boot Project)	26

Projects Download/Login Method without Project Differences	26
Monitoring	26
Debugging.....	27
Breakpoints	27
Stepping.....	27
Printing.....	28
Security.....	28
Project	28
PLC	28
4. QUICK START	29
Start MasterTool IEC XE.....	29
Adding Modules.....	35
Creating POUs.....	37
Creating Tasks.....	39
Task Configuration.....	41
POU – Task Connection.....	43
Maximum Number of Tasks	43
CPU Configuration.....	44
Libraries.....	44
Inserting a Protocol Instance.....	45
MODBUS RTU.....	45
MODBUS Ethernet	46
Building a Project.....	48
Simulation Mode.....	50
Create and Run Projects.....	52
Declare Variables in MainPrg.....	52
Enter Programming Code in the Body of MainPrg.....	53
Create a Programming POU (ST function block FB1)	54
Define the Resources for Running and Controlling the Program on Nexto.....	55
Run and Watch the Application on Nexto.....	55
Debug an Application	58
Help.....	58
Context Sensitive Help.....	58
Uninstallation, Update, Repair.....	59
5. USER INTERFACE.....	60
User Interface Components.....	60
Windows, Views, Editor Windows	61
Customizing User Interface.....	62
Arranging Menu Bars and Tool Bars	62
Zoom	62
User Interface in Online Mode.....	62
Standard Menus and Commands.....	64
Standard Menus and Commands.....	64
User Files Memory	70
6. USER AND ACCESS RIGHT MANAGEMENT.....	71
User and Access Right Management of the Project	71
User Management	71
Access Right Management.....	76
User and Access Right Management of the CPU.....	78
Users and Groups	79

Access Rights	84
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.....	95
Source Upload.....	97
Source Download	97
Print.....	97
Page Setup.....	97
Recent Projects.....	97
Exit.....	98
Edit Menu.....	98
Undo/Redo	99
Clipboard.....	99
Select All.....	101
Find/Replace	101
Browse.....	105
Insert File as Text.....	105
Advanced.....	105
Bookmarks	106
Input Assistant.....	107
Auto Declare... ..	109
Messages View.....	111
View Menu	111
Default Navigators	112
Messages	112
Element Properties	113
Product Library	113
Toolbox	114
Watch List View	114
Watch	114
Breakpoints	114
Call Stack	115
Cross Reference View.....	116
Start Page	117
Full Screen	117
Properties	117
Project Menu.....	124
Add Object	125
Add Device.....	126
Scan Devices	126
Add Folder	127
Edit Object	128
Edit Object with	128
Project Information	128
Project Settings.....	133
Project Update.....	139
Export to CSV	141
Import from CSV	142

Special cases Related to the Import and Export of CSV	144
Export PLCopenXML	145
Import PLCopenXML	147
Document	147
Compare	148
User Management	151
Recipe Menu	153
Menus from the Editors of Programming Languages	153
FBD/LD/IL Editor Commands	154
CFC Commands	170
SFC Commands	178
Menu Text List.....	185
Insert Text	185
Create Global Text List	185
Add Language	185
Remove Language	186
Import/Export Textlists	186
Update Visualization Text IDs	190
Check Visualization Text IDs	190
Remove Visualization Text IDs	190
Menu Visualization.....	190
Interface Editor	191
Hotkeys Configuration	191
Element List	191
Activate Keyboard Usage.....	191
Alignment.....	191
Order	192
Group.....	193
Ungroup.....	193
Background	193
Build Menu.....	194
Generate Code	194
Clean	194
Clean All	194
Online Menu	194
Login	195
Logout	197
Create boot application.....	197
Logoff current online user	197
Download	197
Online Change.....	198
Source download to connected device	198
Redundancy Configuration	199
OPC Configuration.....	199
CPU Informations	199
Reset Warm.....	200
Reset Cold	200
Reset Origin	200
Simulation	201
Export Online Variables.....	201
Import Online Variables.....	204
Debug Menu.....	205
Start	205
Stop	205
Breakpoints	206
New Breakpoint	206

Toggle Breakpoint.....	209
Step Over.....	209
Step Into.....	209
Step Out.....	210
Run to Cursor.....	210
Set Next Statement.....	210
Show Next Statement.....	211
Write Values.....	211
Force Values.....	211
Unforce Values.....	214
Add All Forces to the Watch List.....	215
Display Mode.....	215
Tools Menu.....	215
Library Repository.....	215
Install Library.....	219
Device Repository.....	220
Install Device.....	221
Options.....	221
Window Menu.....	245
Next Editor.....	246
Previous Editor.....	246
Close Editor.....	246
Close All Editors.....	246
Reset Window Layout.....	246
New Horizontal Tab Group.....	246
New Vertical Tab Group.....	246
Float.....	247
Dock.....	247
Auto Hide.....	247
Next Pane.....	247
Previous Pane.....	247
Window <n>.....	247
Windows.....	247
Help Menu.....	248
Contents.....	248
Index.....	248
Search.....	248
Contact Support.....	249
Update Software License.....	249
Documentation.....	249
Altus Home Page.....	249
About.....	249
Bus Editor Menu.....	249
Zoom out.....	250
Zoom in.....	250
Rulers.....	250
Ports.....	250
Grid.....	250
Declaration.....	250
Tabular Declaration Editor - Declaration.....	250
Edit Declaration Header.....	250
Insert.....	250
Move Down.....	250
Move Up.....	250
Trace.....	251

8. EDITORS	252
General Regards on Editors	252
Bus Editor	252
Add Device.....	253
Remove Device	254
Digital I/O Module Editor	254
Process Data.....	254
Module Parameters.....	255
Bus: I/O Mapping.....	256
Bill of Materials	256
PROFIBUS Modules Base – Ponto Series	257
Configuration and Consumption	258
Programming Language Editors	259
Declaration Editor	259
Textual Declaration Editor	259
Tabular Declaration Editor.....	260
Declaration Editor in Online Mode	262
Device Editor	263
Communication Settings	263
Files.....	269
Log	270
OPC Communication Editors	271
OPC Configuration.....	271
Symbol Configuration Object	274
Testing OPC Communication Using Simulation	275
DUT Editor	275
FBD/LD/IL Editor	276
Global Variables List Editor	276
Library Manager Editor	277
Library Manager Editor Window	277
Libraries Menu	280
Recipe Manager Editor	280
Adding the Object	280
Storage Tab Fields.....	282
General Tab Fields	283
Recipe Definition Editor	284
Add Object	284
Columns	286
Actions and Context Menus	287
Task Editor	291
Task Configuration.....	291
Task Editor, Usage	292
Task Editor in Online Mode.....	297
Task Configuration Commands	298
Trace Editor	298
Trace.....	298
Shortcuts for Trace.....	302
Trace Editor.....	302
Add Variable	305
Configuration	306
Appearance.....	313
Trace Editor in Online Mode	314
Watch List Editor	315
Watch View / Watch List Editor	315
Create Watch List.....	315

Watch List in Online Mode	316
MODBUS Editor	317
MODBUS RTU Master for Direct Representation (% Q)	317
MODBUS RTU Master for Symbolic Mapping	321
MODBUS RTU Slave for Direct Representation (% Q)	324
MODBUS RTU Slave for Symbolic Mapping.....	325
MODBUS Ethernet Client for Direct Representation (% Q)	327
MODBUS Ethernet Client for Symbolic Mapping	330
MODBUS Ethernet Client for Direct Representation (% Q)	334
MODBUS Ethernet Server for Symbolic Mapping	337
PROFIBUS Editor.....	338
CPU Editor.....	338
Serial Interfaces.....	339
Ethernet Interfaces.....	340
PID Control Editor.....	341
Insert PID Control Object in the Application.....	341
Graphical Environment	342
Visualization.....	354
Overview	355
Visualization Commands	357
Visualization Editor	359
Visualization Object.....	374
Visualization Elements.....	390
Visualization Libraries	481
Visualization Manager with Clients	482
Visualization in Online Mode	494
9. INSTALLATION	495
10. DIAGNOSTICS	502
Diagnostics Page	502
Diagnostics	503
11. GLOSSARY	507

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.

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	Nexto Series – Features and Configuration	English
CT114000	Série Nexto – Características e Configurações	Portuguese
CS114000	Serie Nexto – Especificaciones y Configuraciones	Spanish
CE114100	CPUs Nexto Series – Features and Configuration	English
CT114100	UCPs Série Nexto – Características e Configurações	Portuguese
CS114100	UCPs Serie Nexto – Especificaciones y Configuraciones	Spanish
CE103705	MasterTool IEC XE – Features and Configuration	English
CT103705	MasterTool IEC XE – Características e Configurações	Portuguese
CS103705	MasterTool IEC XE – Especificaciones y Configuraciones	Spanish
MU214600	Nexto Series User Manual	English
MU214000	Manual de Utilização Série Nexto	Portuguese
MU214300	Manual Del Usuario Serie Nexto	Spanish
MU214605	Nexto Series CPUs User Manual	English
MU214100	Manual de Utilização UCPs Série Nexto	Portuguese
MU214305	Manual del Usuario UCPs Serie Nexto	Spanish
MU299609	MasterTool IEC XE User Manual	English
MU299048	Manual de Utilização MasterTool IEC XE	Portuguese
MU299800	Manual del Usuario MasterTool IEC XE	Spanish
MP399609	IEC 61131 Programming Manual	English
MP399048	Manual de Programação IEC 61131	Portuguese
MP399800	Manual de Programación IEC 61131	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 e-mail 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
- **Basic:** software that allows the programming and project loading of up to 2048 I/O points
- **Professional:** programming software for all Nexto Series CPUs
- **Advanced:** programming software with tools for large scale applications using half-cluster redundancy

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

	Lite	Basic	Professional	Advanced
Free version	Yes	No	No	No
Available languages	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 Controllers				
XP300	Yes	Yes	Yes	Yes
XP315	Yes	Yes	Yes	Yes
XP325	Yes	Yes	Yes	Yes
NX3003	Yes	Yes	Yes	Yes
NX3004	Yes	Yes	Yes	Yes
NX3005	Yes	Yes	Yes	Yes
NX3010	Yes	Yes	Yes	Yes
NX3020	No	No	Yes	Yes
NX3030	No	Yes	Yes	Yes
NX5100	Yes	Yes	Yes	Yes
NX5101	Yes	Yes	Yes	Yes
Rack expansion support	No	Yes	Yes	Yes
Rack expansion redundancy support	No	No	Yes	Yes
Ethernet expansion support	No	No	Yes	Yes
Ethernet expansion redundancy support	No	No	Yes	Yes
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

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

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 of Version 1.29 with Other Products

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 of Version 1.40 with Other Products

MasterTool IEC XE versions prior to version 2.01 don't allow the configuration of the Machine Profile. It's necessary to use products of the Nexto Series with software version, or product revision, as in the table below, for this profile to work properly.

	Software version	Product Revision
NX1001	1.2.0.2 or higher	AE or higher
NX1005	1.2.0.3 or higher	AD or higher
NX2001	1.2.0.2 or higher	AD or higher
NX2020	1.2.0.2 or higher	AE or higher
NX5000	1.2.0.1 or higher	AL or higher
NX5001	1.2.0.6 or higher	AP or higher
NX6000	1.2.0.2 or higher	AK or higher
NX6100	1.2.0.1 or higher	AI or higher

Table 2-4. Compatibility of Machine Profile with other Products

ATTENTION:

Due to the compiler update between version 2.09 and 3.00 of MT8500, source codes loaded into UCPs using the MT8500 lower than 2.09, when read by a version 3.00 or higher, can not be connected online without needing a new download.

Minimum and Recommended Requirements

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

MasterTool IEC XE	
Platform	PC with operating system: Until version 3.05: Windows XP® (32bits), Windows Vista® (32 bits), Windows 7 SP1® (32bits or 64bits) or Windows 8.1® (64 bits) From version 3.10: Windows 7 SP1® (32bits or 64bits) or Windows 8.1® (64 bits)
Processor	Intel Core 2 Duo 1.66 GHz (minimum)
Disc Space	1 Gbyte (minimum), 2 Gbytes (recommendable)
RAM Memory	2 Gbytes (minimum), 8 Gbytes (recommendable)
Resolution	1024 x 768 (recommendable)
Language	Any language

Table 2-5. Minimum Requirements

NOTES:

- Platform: MasterTool IEC XE installer for Windows 7® 64 bits is available since version 1.20 and for Windows 8.1® 64 bits is available since version 2.00.
- Platform and RAM: Despite the commercialization of computers with more than 3 Gbytes of RAM and 32-bits operating systems, the total memory can only be accessed through a 64-bits operating system. Due to this, it's recommended that a 64-bits operating system is.
- Requirements: As a general rule, PCs with the minimum requirements can be used for non-redundant projects, however for redundant projects, PCs with the recommended requirements should be used.

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 2-6. 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 POU's 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

Project Profiles

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

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

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

MasterTool IEC XE provides several templates compatible with each profile defined for the RTS. When the user selects a template as a model in project creation, the new application will be developed according to a certain profile, adopting the rules, characteristics and standards defined by the profile associated to the template. For further information about each of these profiles consult the Nexto Series CPUs User Manual – MU214605, on chapter Initial Programming.

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

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 POUs, external files, etc. Besides that, it shows configurations objects and project information.

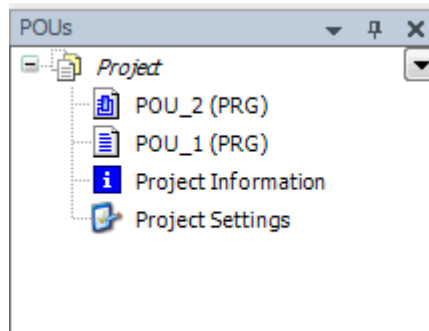


Figure 3-1. POUs Window

Project Settings and Project Information

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


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

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

For a detailed description see **Project Menu**.

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 POU's 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 POU's, Global Variable lists and Library Manager can - instead of being managed as project-globally instantiable units in the POU's 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:

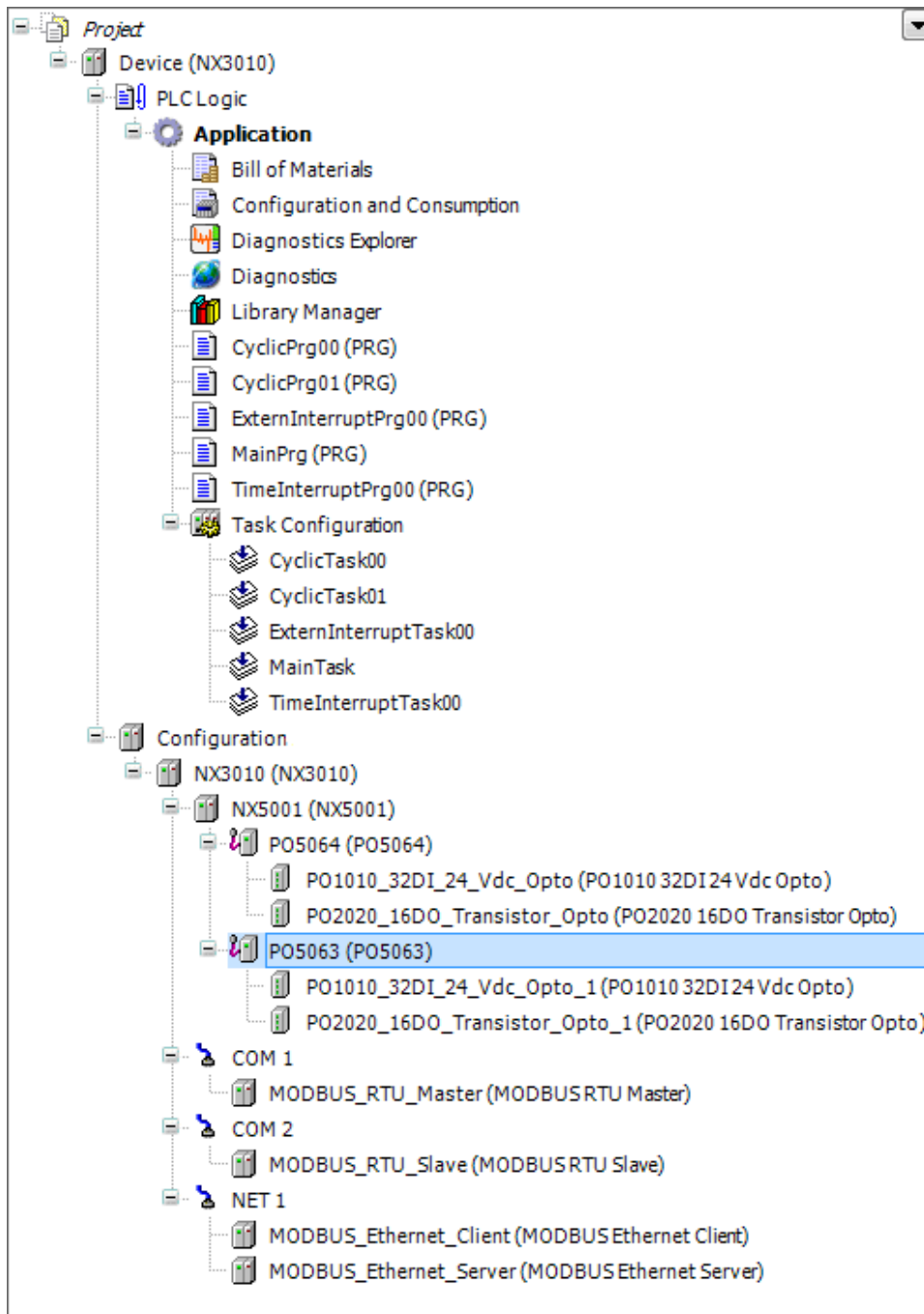




Figure 3-2. Device Tree

- 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 3-1. 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 “children”.

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

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.

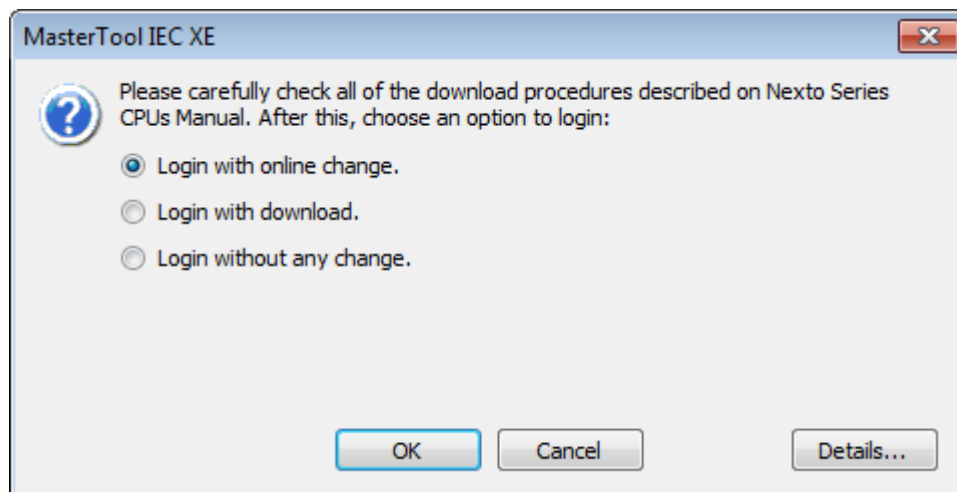


Figure 3-3. Login Dialog

- *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.

ATTENTION:

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

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

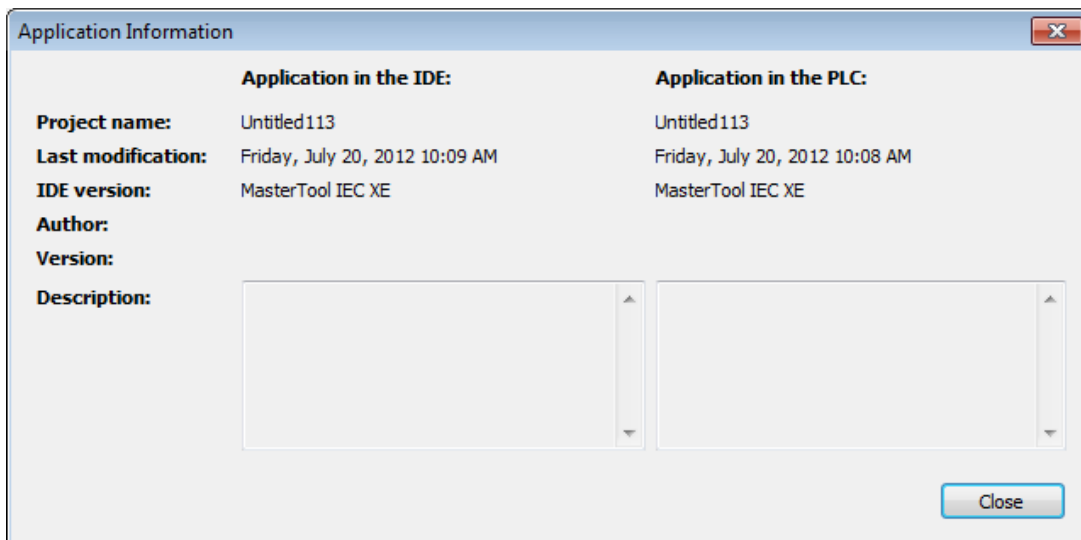


Figure 3-4. Application Information Dialog

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 POU's 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 POU's 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** and **Access Rights** to get information on the respective user and possibilities of management rights in the device Configuration editor.

4. Quick Start

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

The following items will be explored:

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

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.

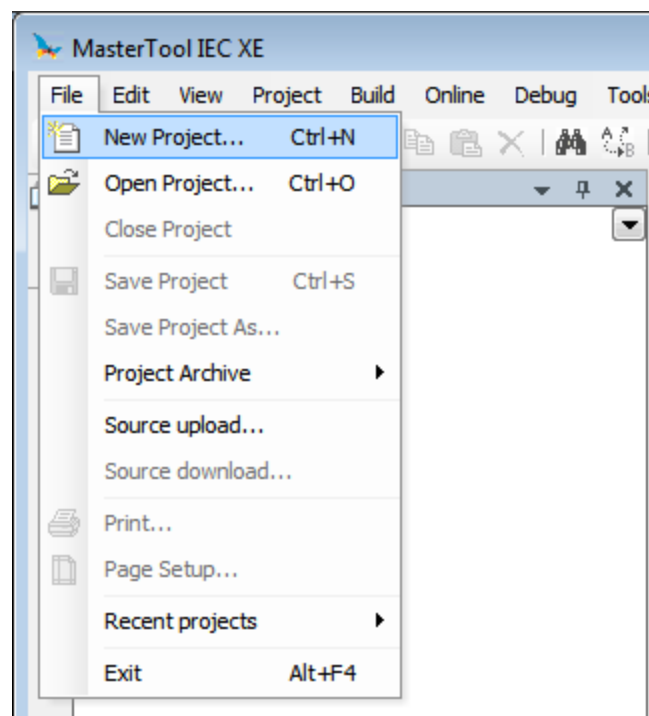


Figure 4-1. New Project

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.

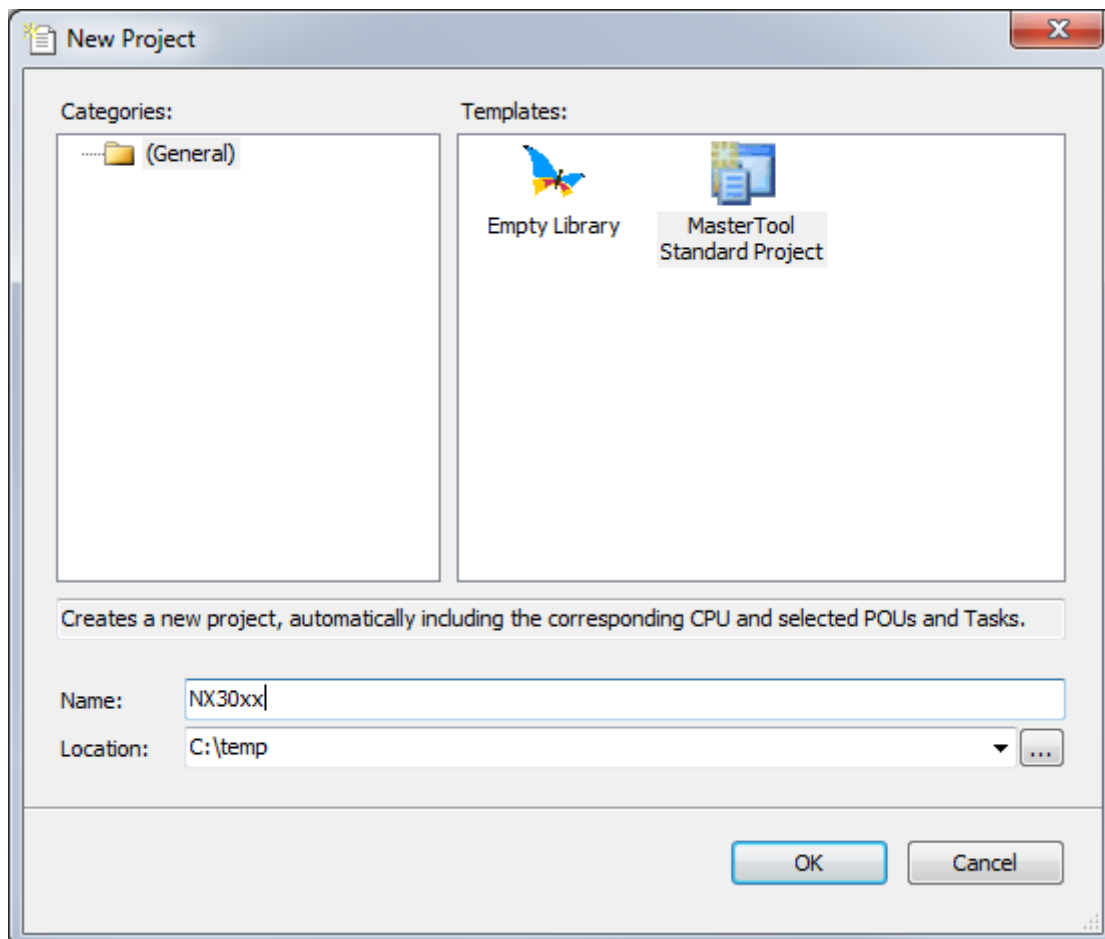


Figure 4-2. Project Classification

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 NX3030 CPU will be used, an NX9000 rack and an NX8000 power supply and the options for Half-Cluster and bus expansion Without Redundancy will be used.

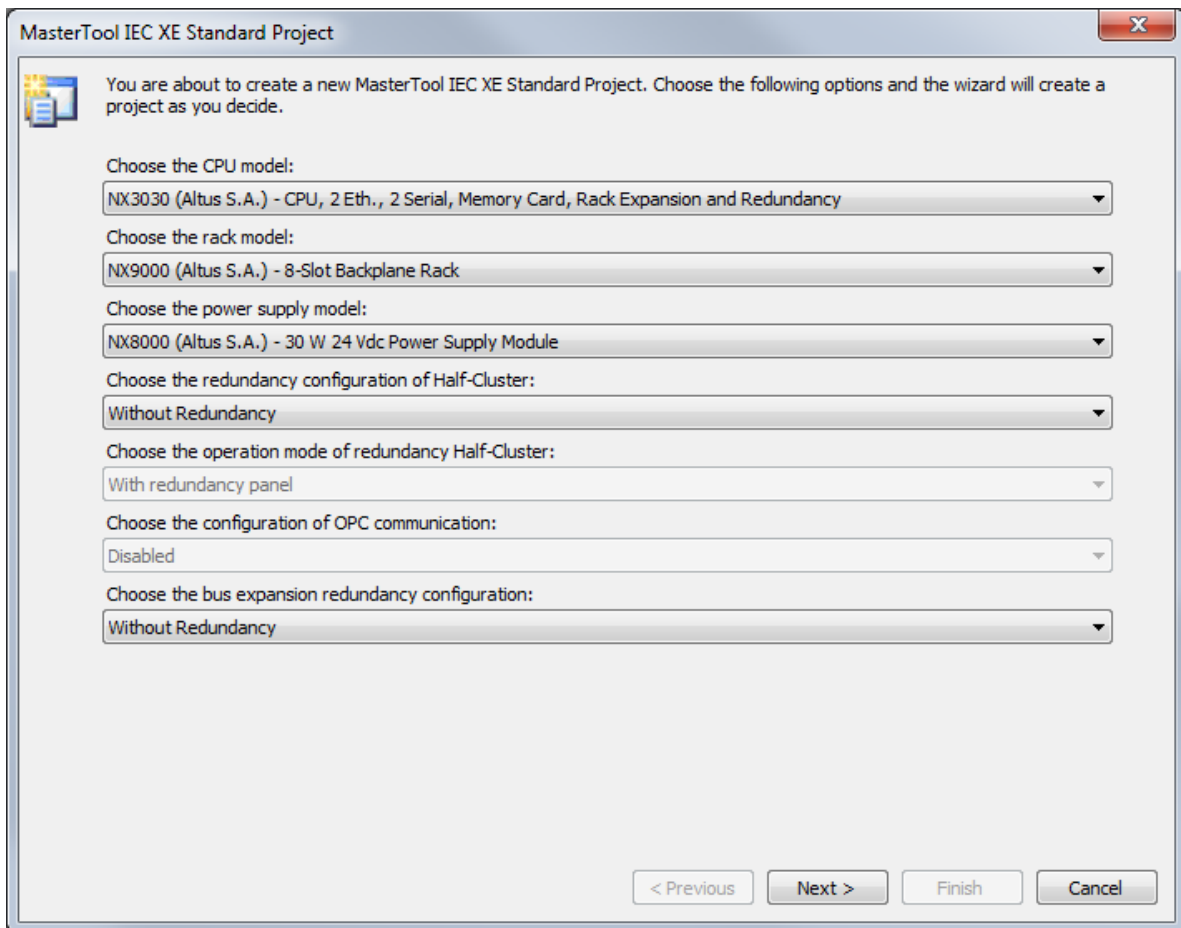
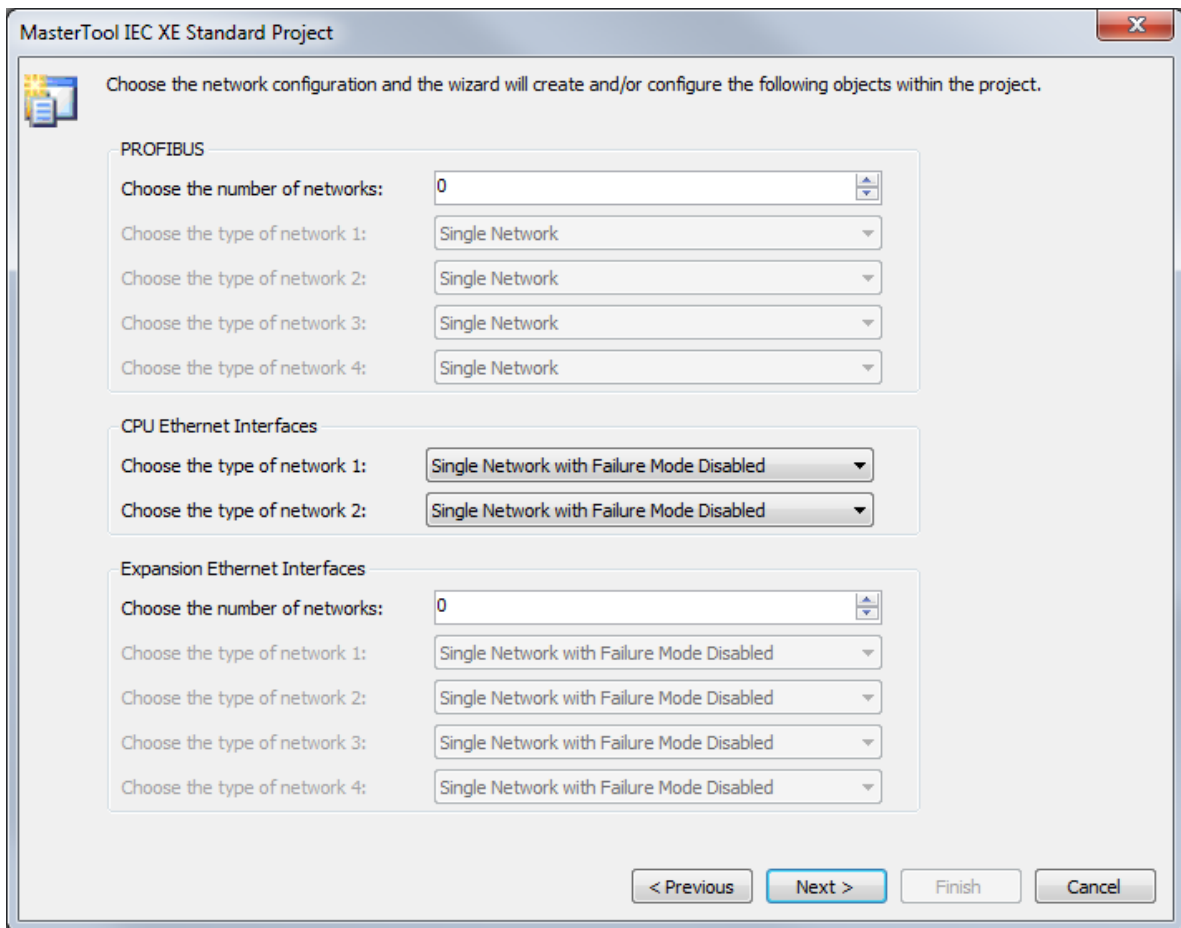


Figure 4-3. Hardware Modules

The next step is to choose the network options. In this page it's possible to choose the quantity of PROFIBUS and Ethernet networks. It's also possible to define if these communication networks are going to be simple or redundant. In the Ethernet network case, it's also possible to define if they are going to trigger a Switchover in case of failure when configured with Half-Cluster redundancy. In this example it wasn't created any networks besides the CPU Ethernet network interfaces.



The screenshot shows a software wizard window titled "MasterTool IEC XE Standard Project". The window contains a header with a close button (X) and a main instruction: "Choose the network configuration and the wizard will create and/or configure the following objects within the project." Below this, there are three sections of configuration options:

- PROFIBUS:** A group box containing:
 - "Choose the number of networks:" with a spin box set to 0.
 - "Choose the type of network 1:" with a dropdown menu set to "Single Network".
 - "Choose the type of network 2:" with a dropdown menu set to "Single Network".
 - "Choose the type of network 3:" with a dropdown menu set to "Single Network".
 - "Choose the type of network 4:" with a dropdown menu set to "Single Network".
- CPU Ethernet Interfaces:** A group box containing:
 - "Choose the type of network 1:" with a dropdown menu set to "Single Network with Failure Mode Disabled".
 - "Choose the type of network 2:" with a dropdown menu set to "Single Network with Failure Mode Disabled".
- Expansion Ethernet Interfaces:** A group box containing:
 - "Choose the number of networks:" with a spin box set to 0.
 - "Choose the type of network 1:" with a dropdown menu set to "Single Network with Failure Mode Disabled".
 - "Choose the type of network 2:" with a dropdown menu set to "Single Network with Failure Mode Disabled".
 - "Choose the type of network 3:" with a dropdown menu set to "Single Network with Failure Mode Disabled".
 - "Choose the type of network 4:" with a dropdown menu set to "Single Network with Failure Mode Disabled".

At the bottom of the window, there are four buttons: "< Previous", "Next >" (highlighted in blue), "Finish", and "Cancel".

Figure 4-4. Network Options

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

The screenshot shows a window titled "MasterTool IEC XE Standard Project" with a close button in the top right corner. The main area contains the following configuration options:

- A message: "Choose the quantity of I/O points that will be used and the wizard will create the following objects within the project."
- A dropdown menu for "Choose the solution for I/O modules:" with "Nexto" selected.
- Seven numeric input fields, each with up and down arrow buttons, all set to "0":
 - Digital input points:
 - V/I analog input points:
 - Thermocouple analog input points:
 - RTD analog input points:
 - V/I analog output points:
 - Transistor digital output points:
 - Relay digital output points:
- A dropdown menu for "Expansion cable type:" with "Not Connected" selected.
- Summary text:
 - "Quantity of I/O modules that will be used: 0 of 128."
 - "Quantity of racks that will be used: 1 of 25."
- Navigation buttons at the bottom: "< Previous", "Next >" (highlighted in blue), "Finish", and "Cancel".

Figure 4-5. I/O Points Configuration

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.

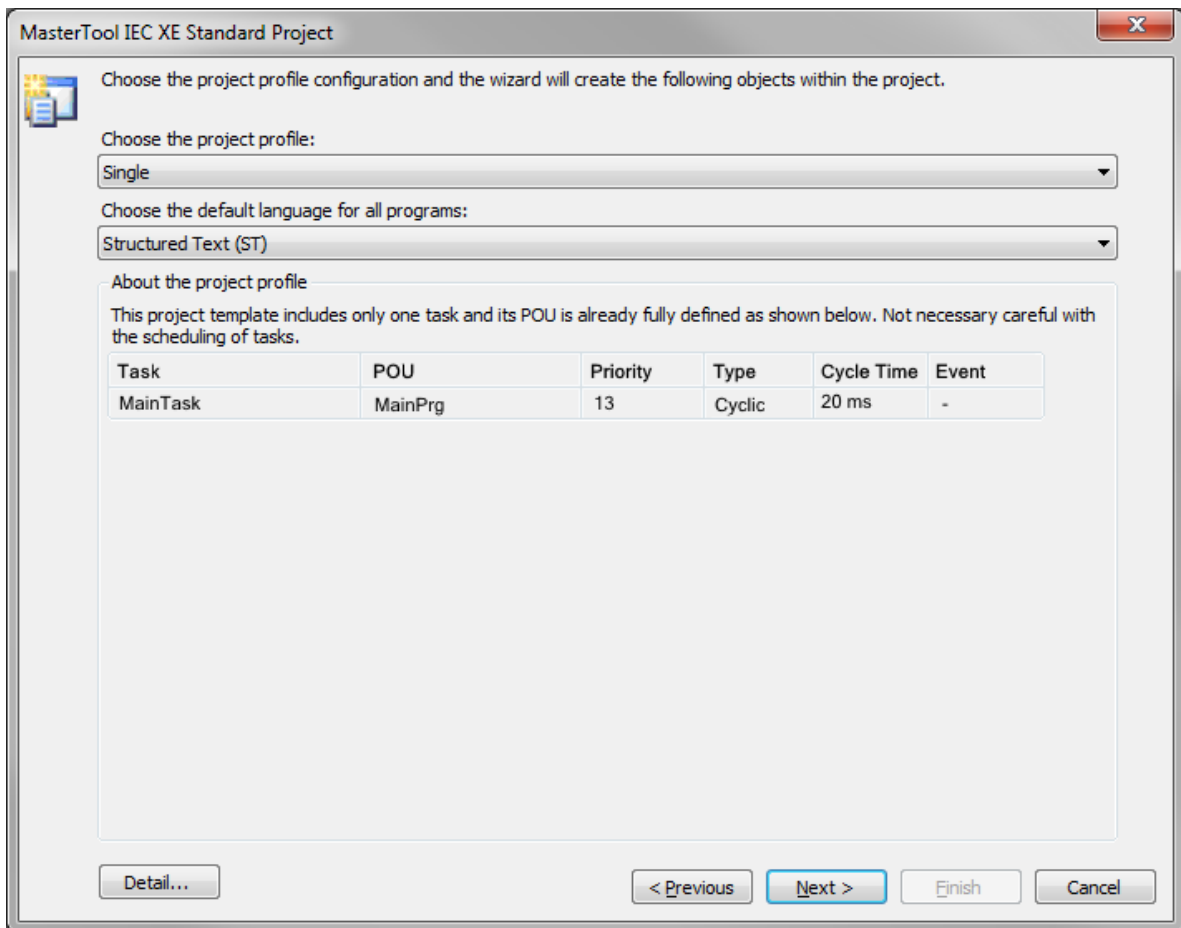


Figure 4-6. Profile Selection

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.

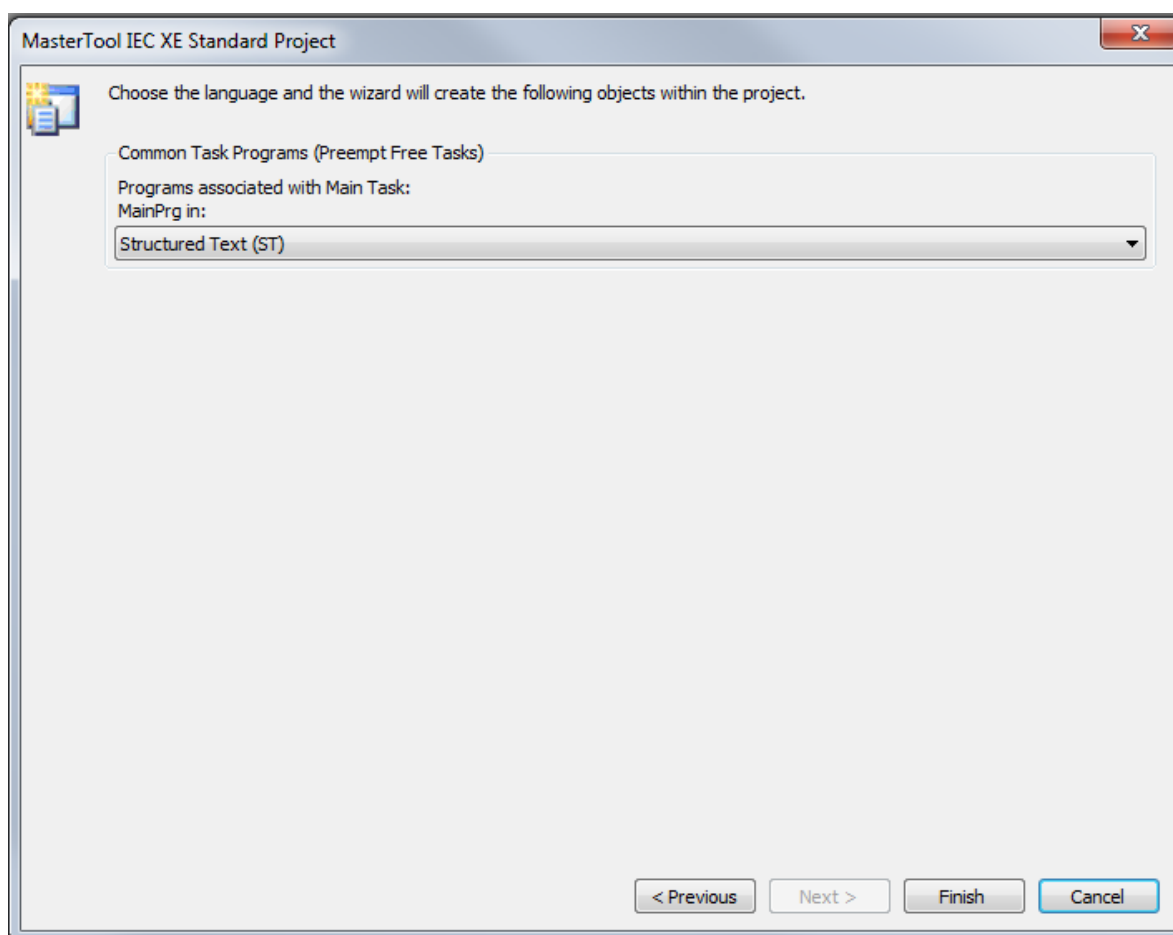


Figure 4-7. Programming Language

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 4-8:

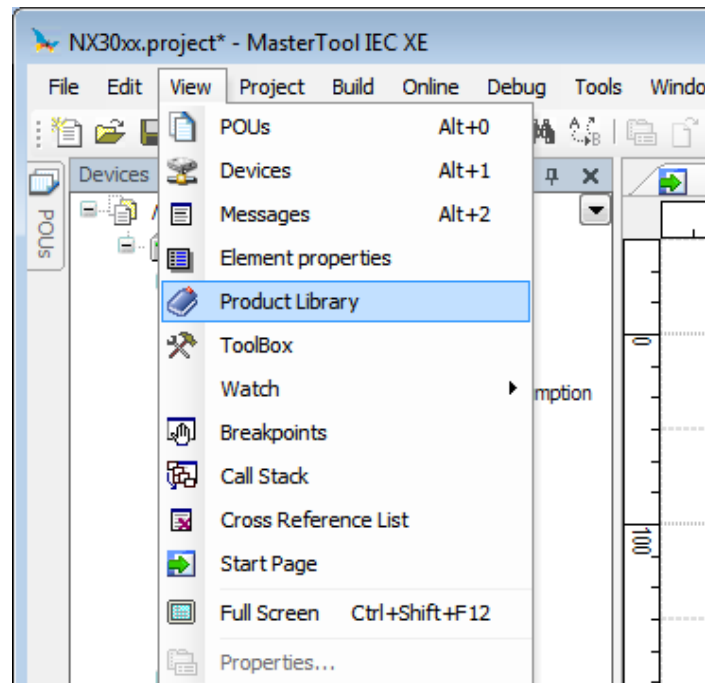


Figure 4-8. Library Visualization

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

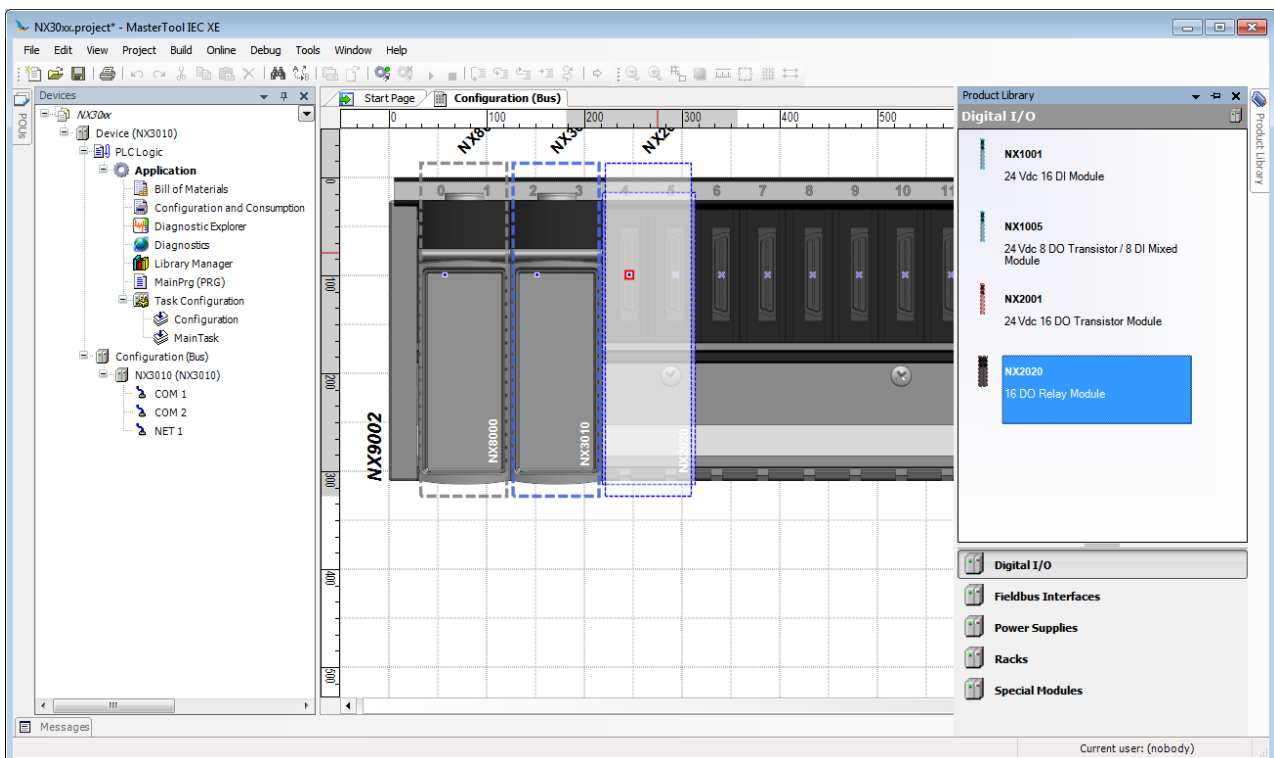


Figure 4-9. Adding Modules

Creating POUs

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

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

To 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 4-10.

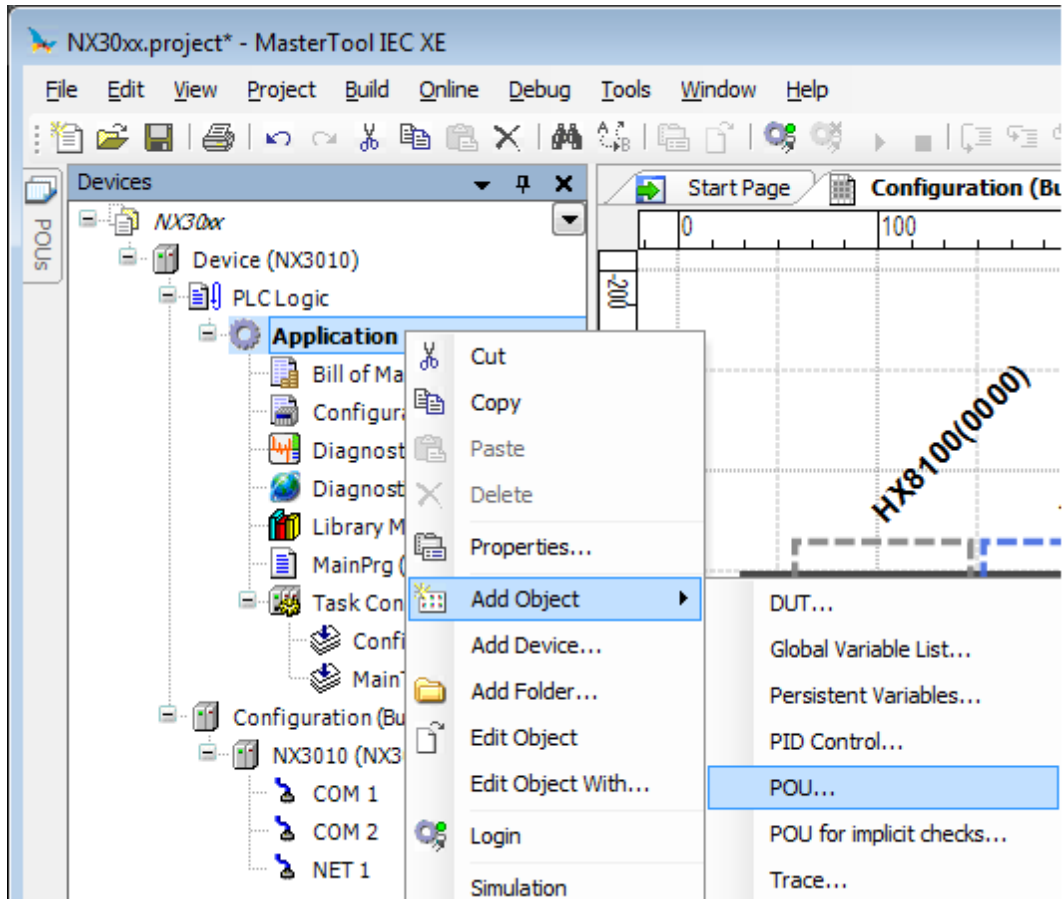


Figure 4-10. Inserting POUs

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

Add POU ✕

Create a new POU (Program Organization Unit)

Name: POU

Type:

Program

Function Block

Extends: ...

Implements: ...

Method implementation language: Structured Text (ST) ▾

Function

Return type: ...

Implementation language: Structured Text (ST) ▾

Open Cancel

Figure 4-11. 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.

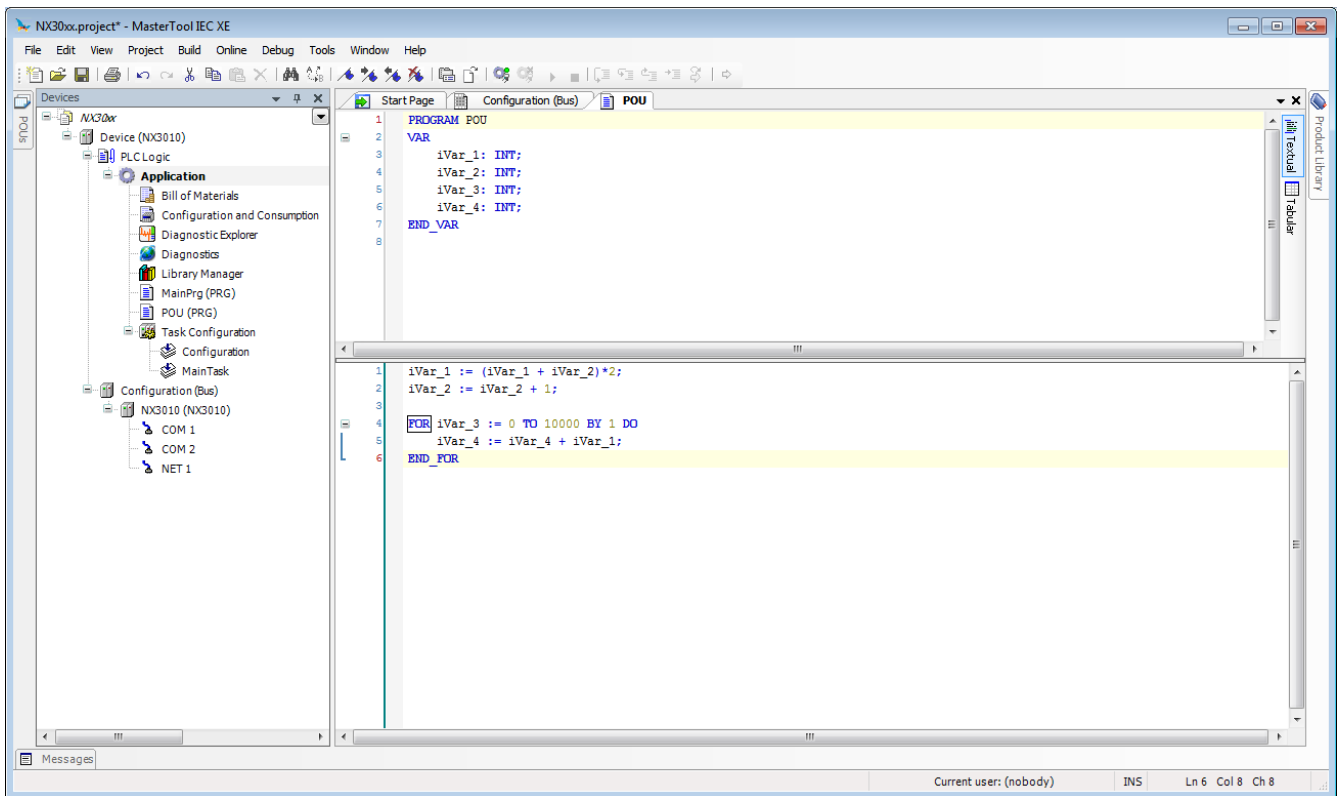


Figure 4-12. POU Editing

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 4-13.

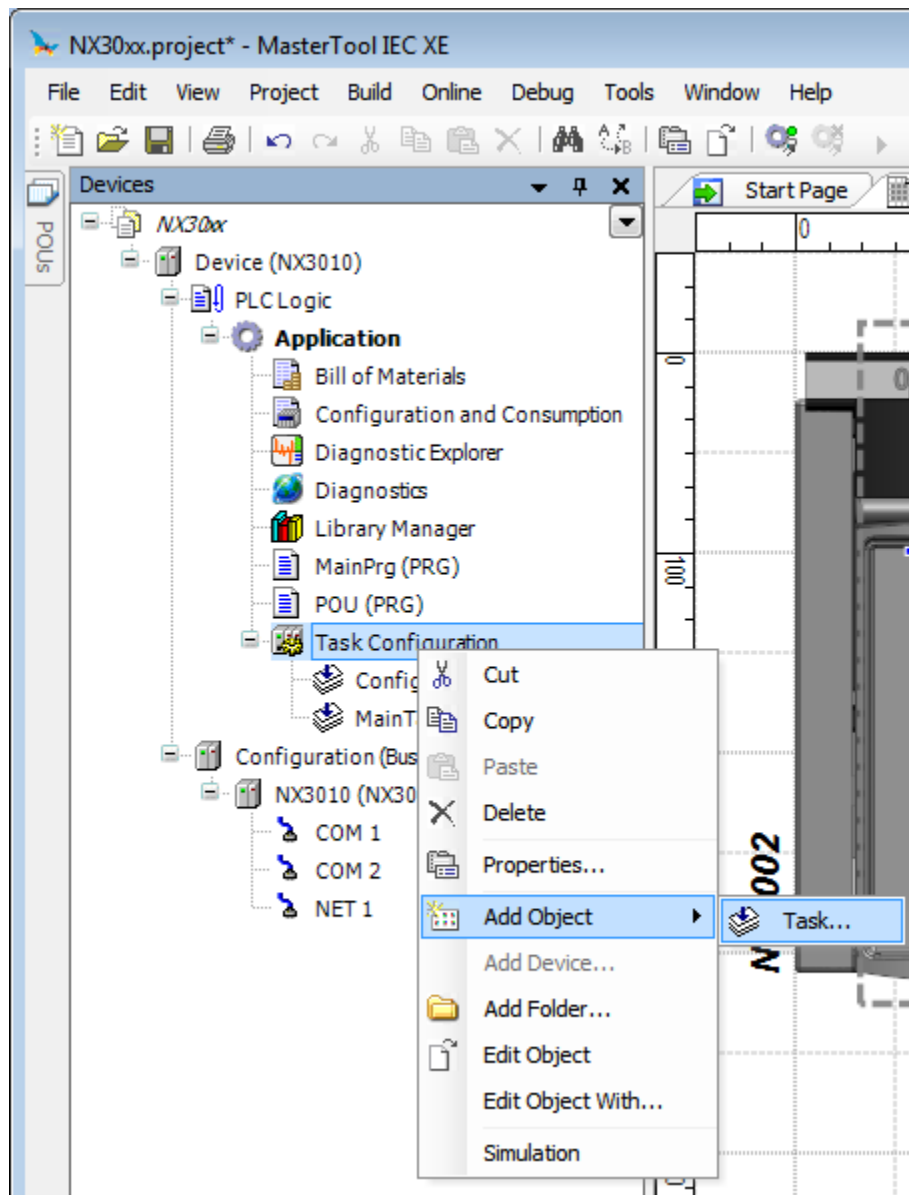


Figure 4-13. Creating a Task

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

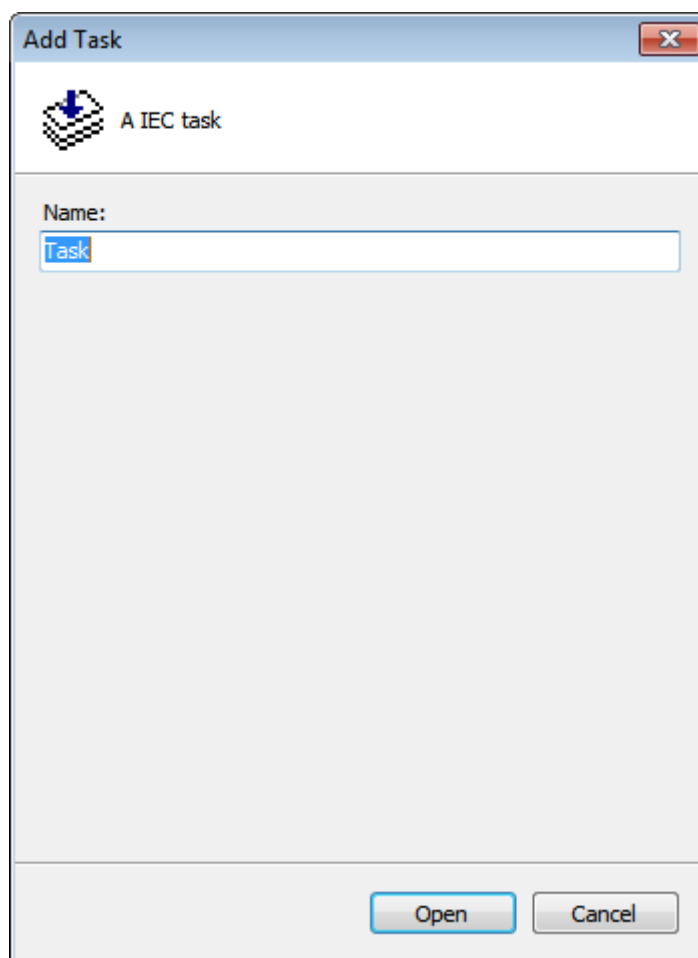


Figure 4-14. Task Name

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. The maximum time for the MainTask in Single, Basic, Normal, Expert and Custom profiles is 750 ms, and the minimum time is 5 ms. In

Machine Profile, the maximum time is 100 ms and the minimum is 5 ms. It's recommended to set the task interval to, at least, twice its cycle (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**.

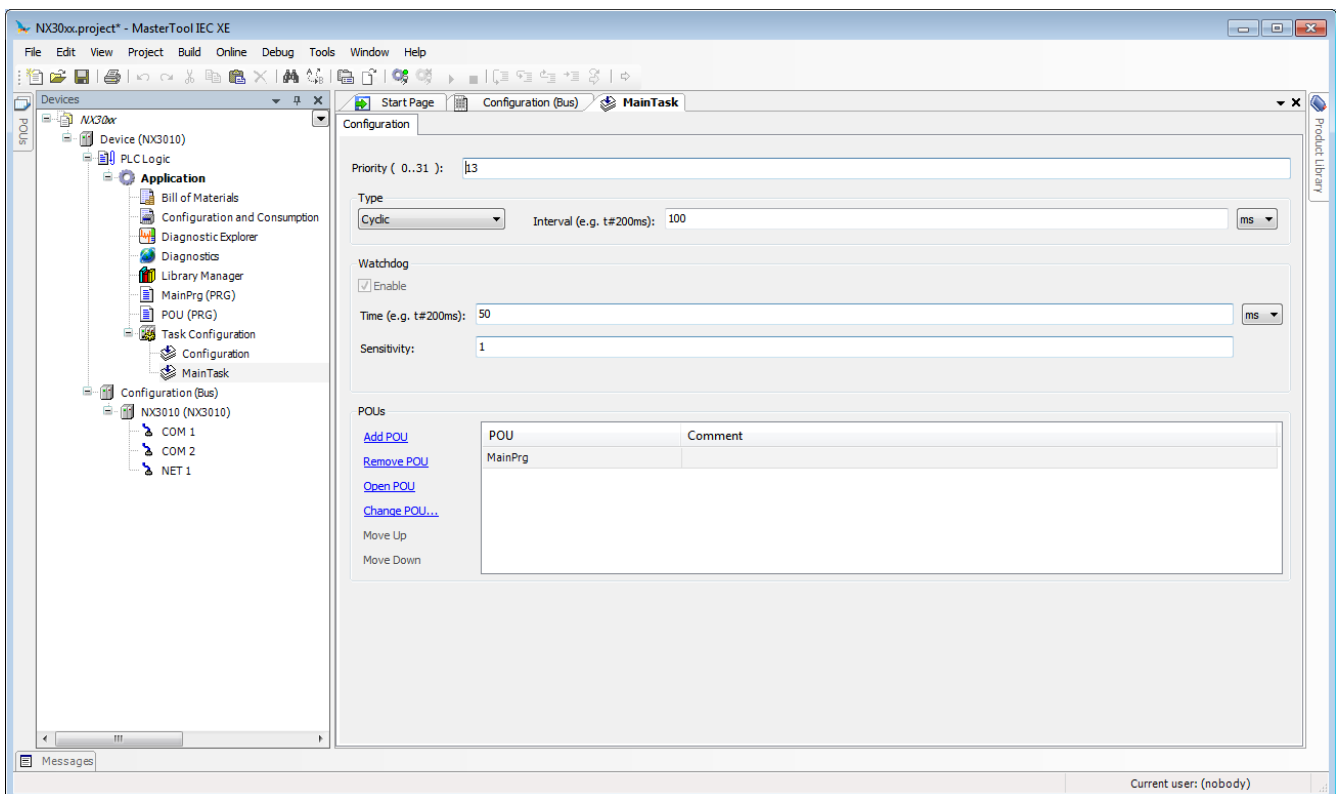


Figure 4-15. Configuring a Task

Below, Table 4-1 shows the verifications performed by MasterTool IEC XE for the Interval field configuration of the cyclic tasks, when the Sensitivity field is 1. For the Custom Profile, the consistency on the task interval and on the watchdog time isn't performed.

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

Table 4-1. Maximum Allowed Configurations

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 POU's are already associated to its respective tasks. However, in case the Custom profile is being used or new POU's 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 4-16.

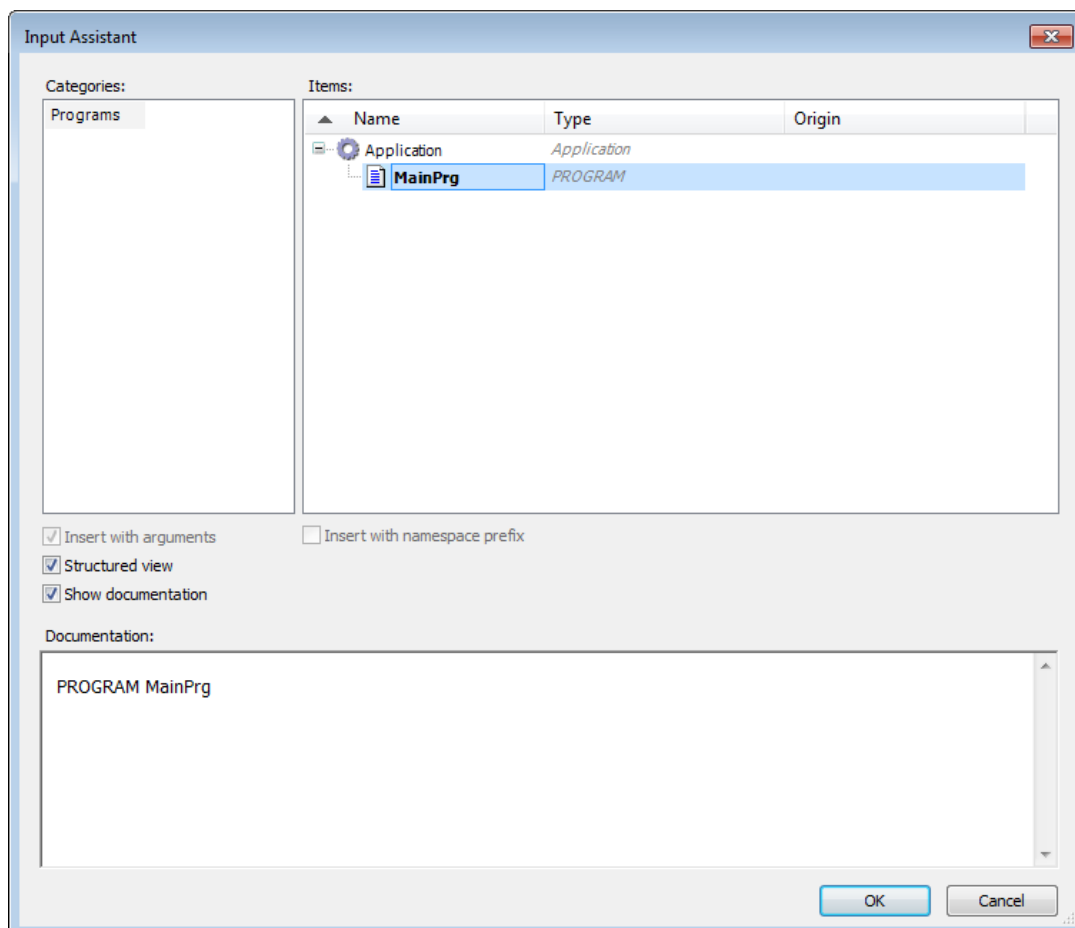


Figure 4-16. Connecting POU's to Tasks

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 Nexto Series CPUs User Manual (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 icon (located in the device tree) and configure the field as described in the **Editors** chapter of this manual and on the Configuration chapter of the Nexto Series CPU User Manual – 214605.

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 4-17.

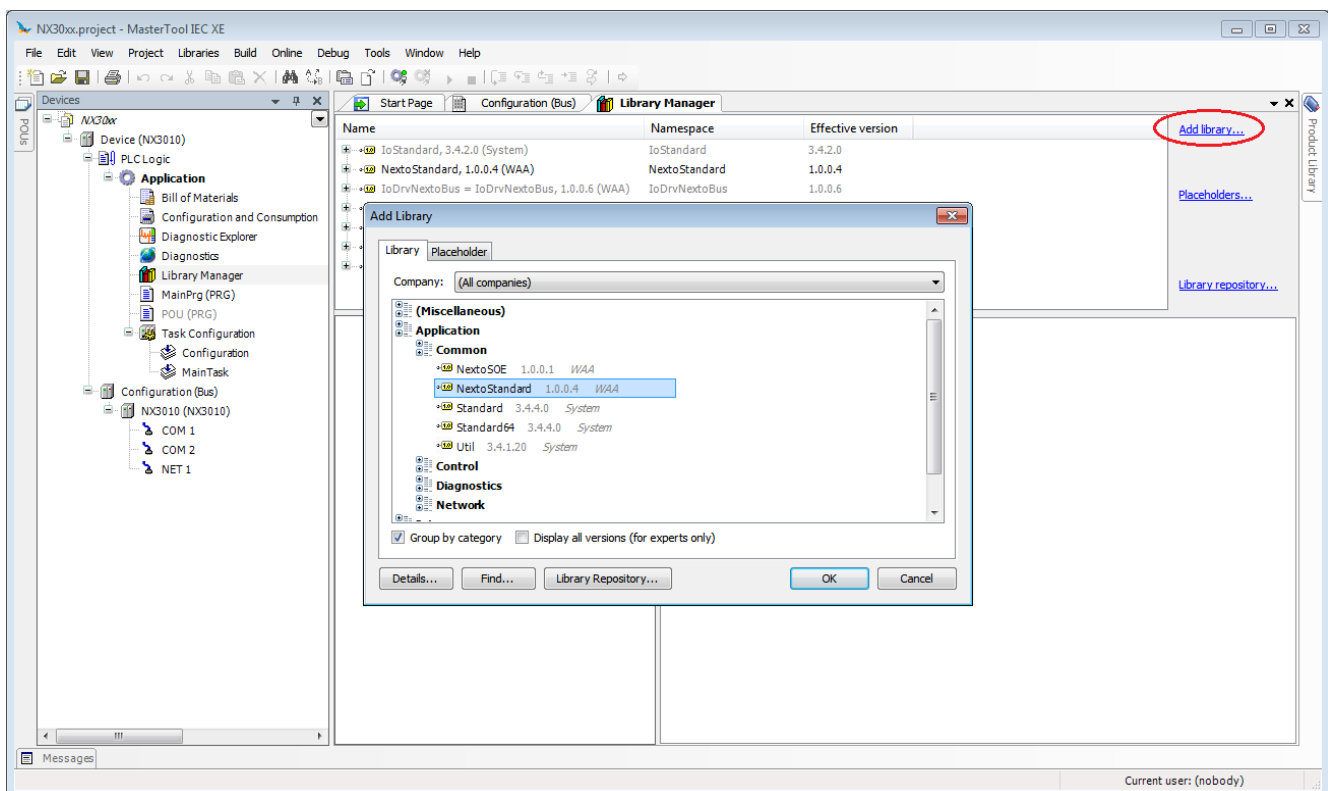


Figure 4-17. Inclusion of a Library in the Project

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 4-18.

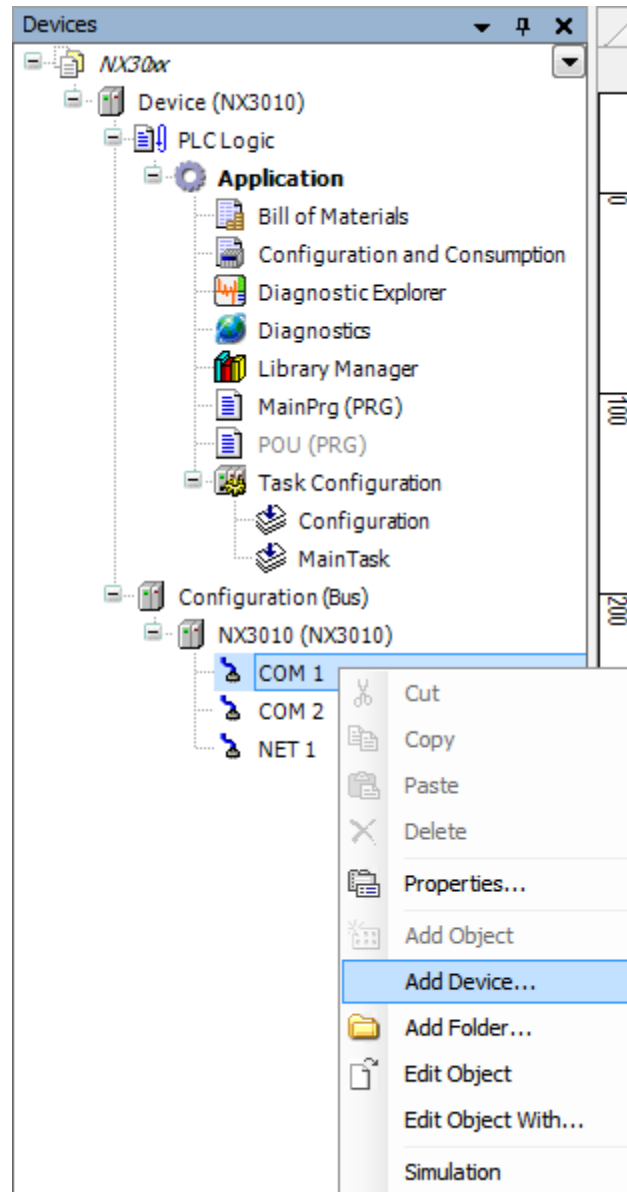


Figure 4-18. Adding an Instance

After that, the available protocols for the user will appear on the screen. Set the mode of the protocol configuration and select *MODBUS Symbol RTU Slave* or *MODBUS RTU Slave* for the correspondent configuration. Then, click *Add Device*, as shown on Figure 4-19.

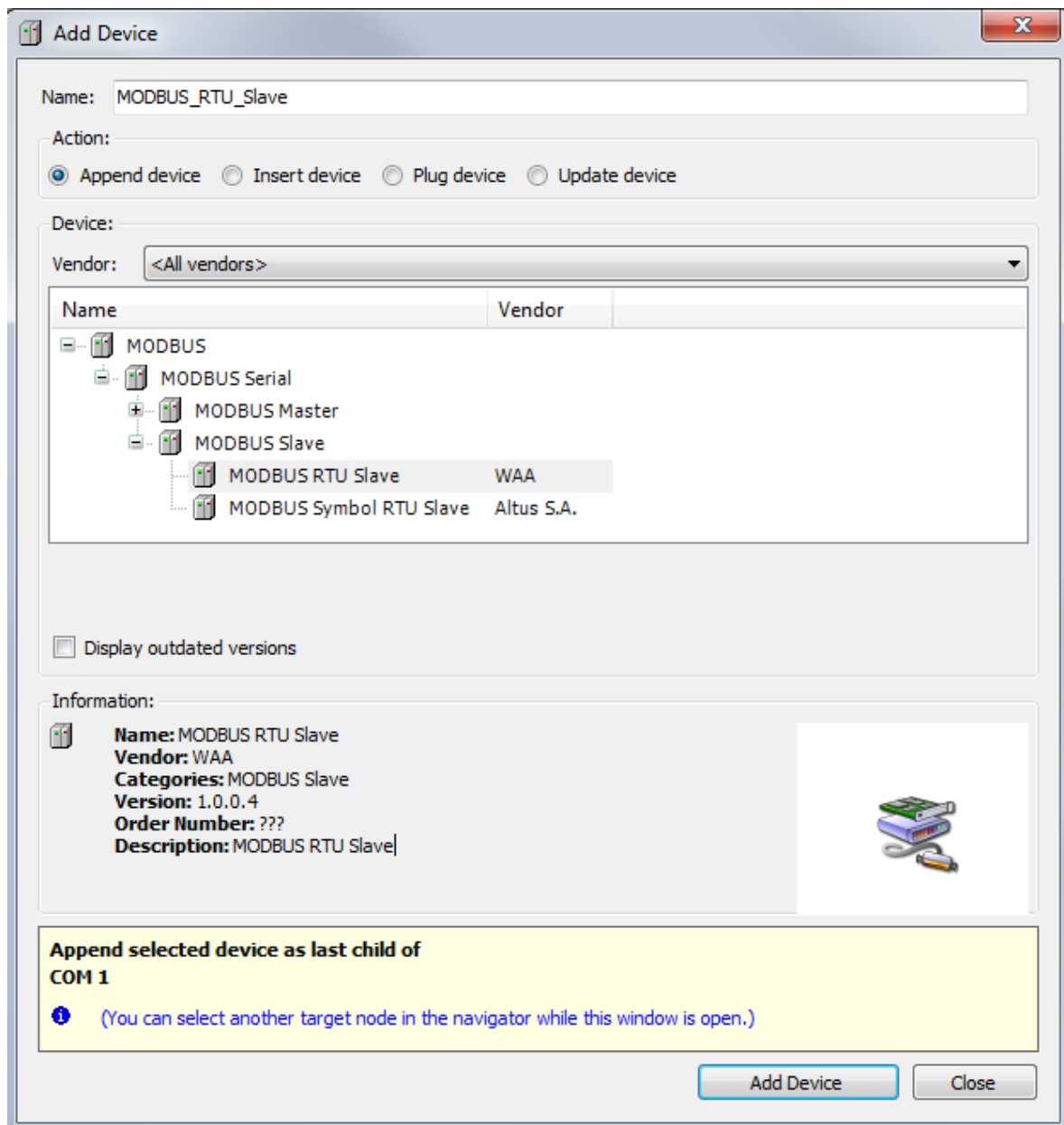


Figure 4-19. Selecting the Protocol

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 4-20.

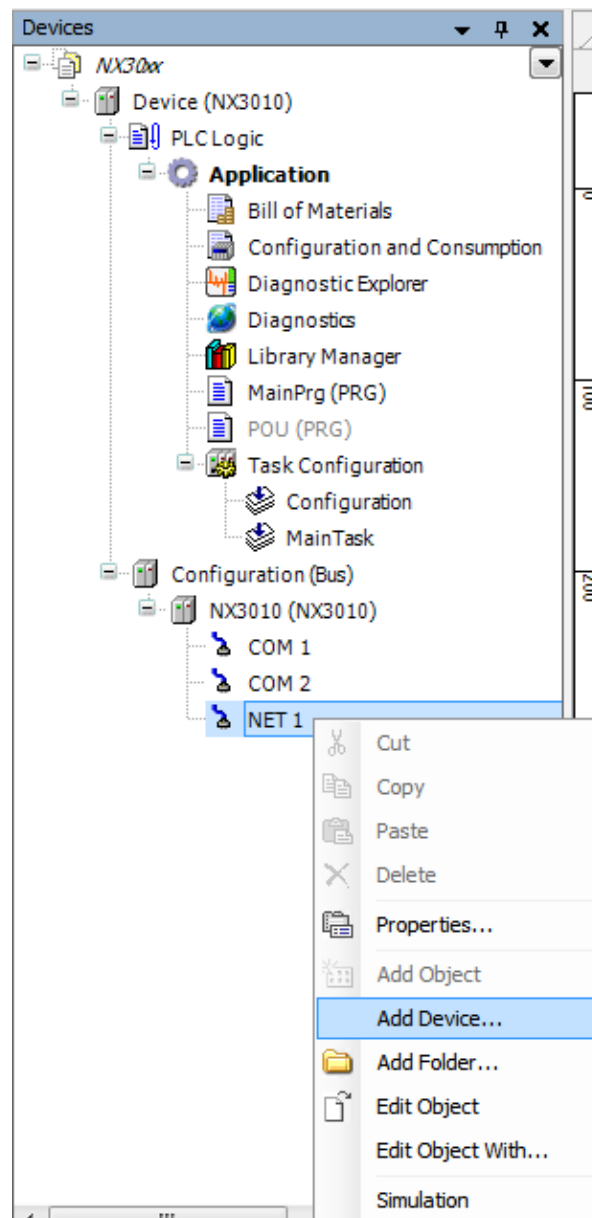


Figure 4-20. Adding an Instance

After that, the available protocols for the user will appear on the screen. . Set the mode of the protocol configuration and select *MODBUS Symbol Client* or *MODBUS Client* for the correspondent configuration. Then, click *Add Device*, as shown on Figure 4-21.

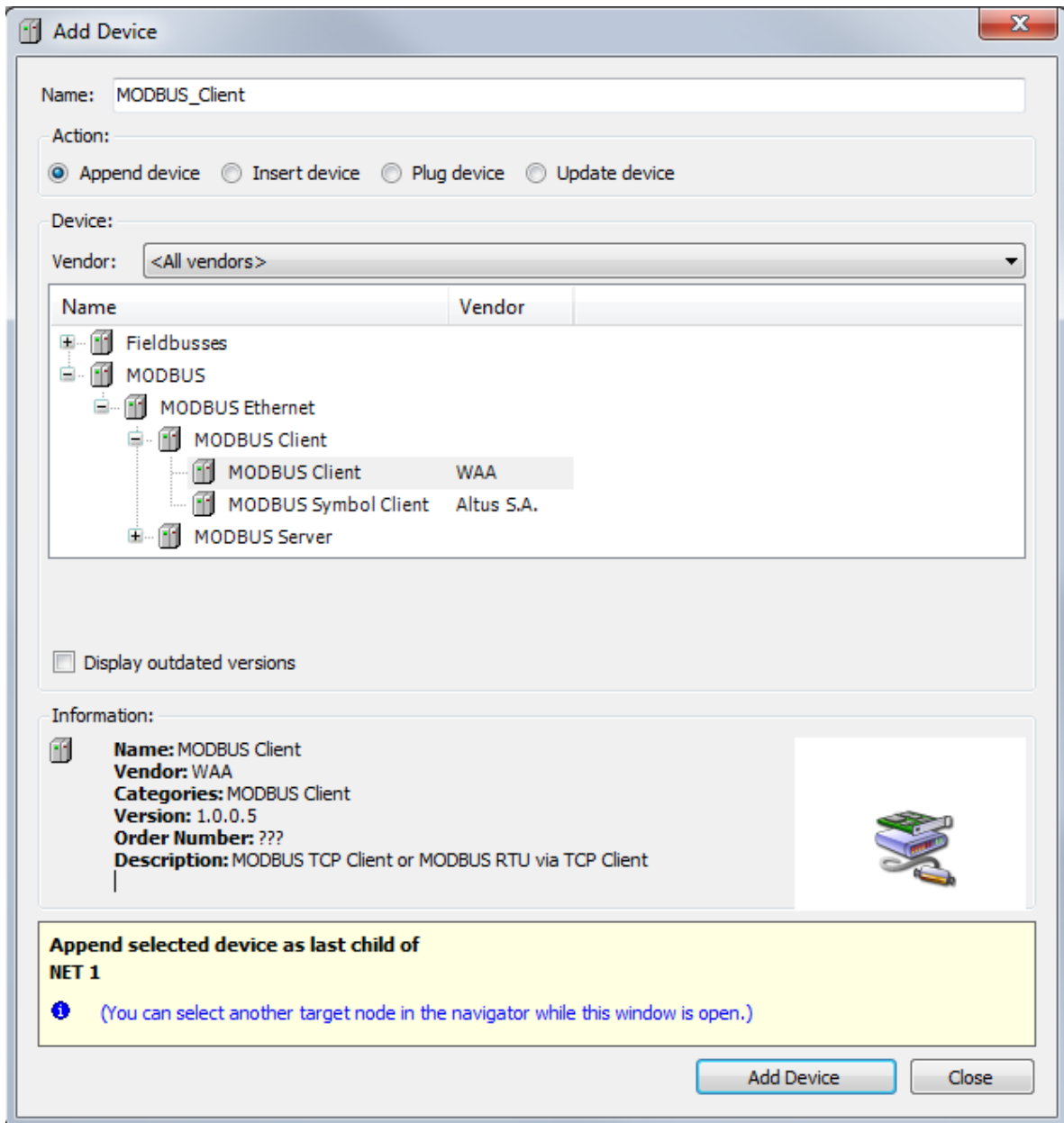


Figure 4-21. Selecting the Protocol

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*.

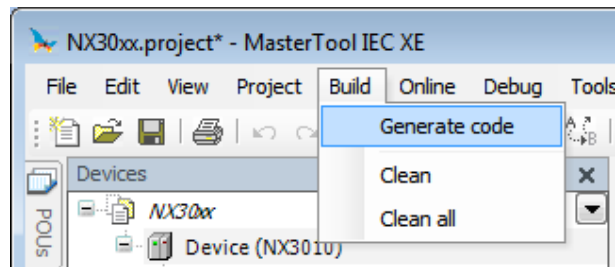


Figure 4-22. Building the Project

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 4-23.

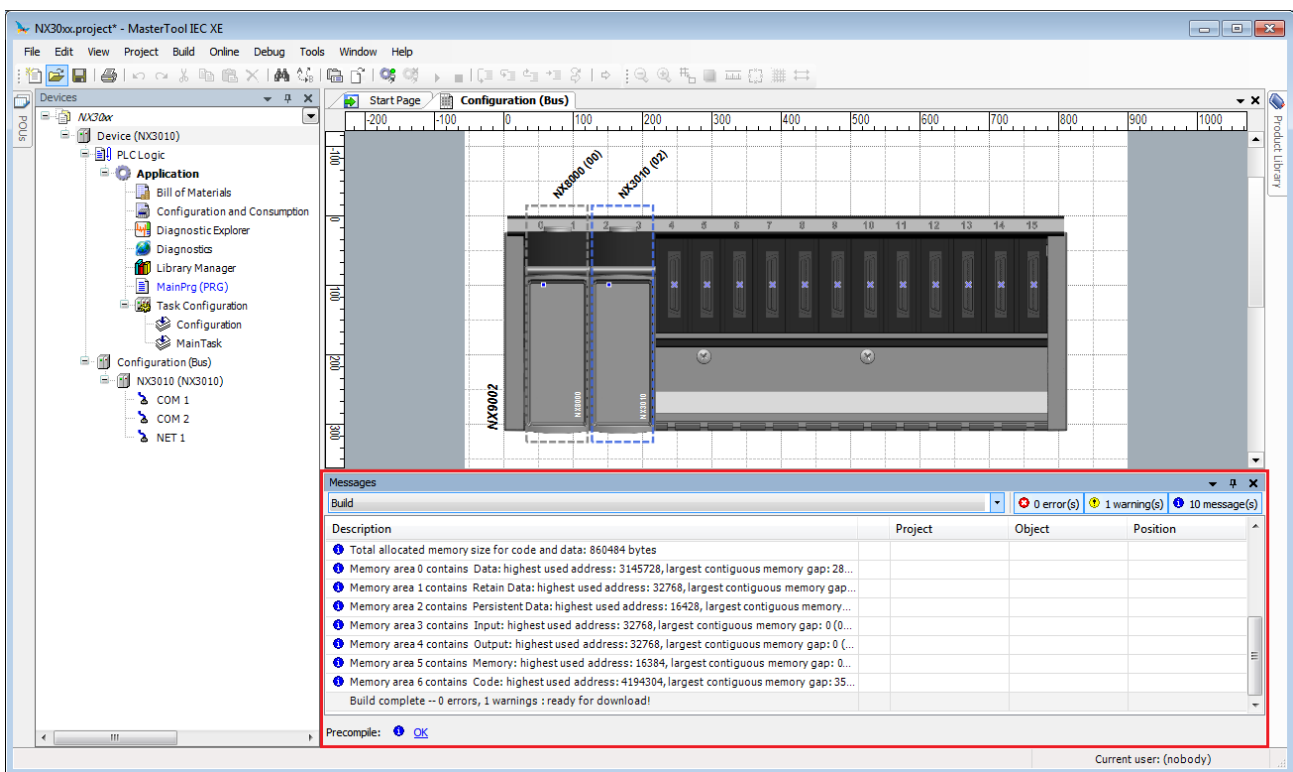


Figure 4-23. Building Messages

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 4-24.

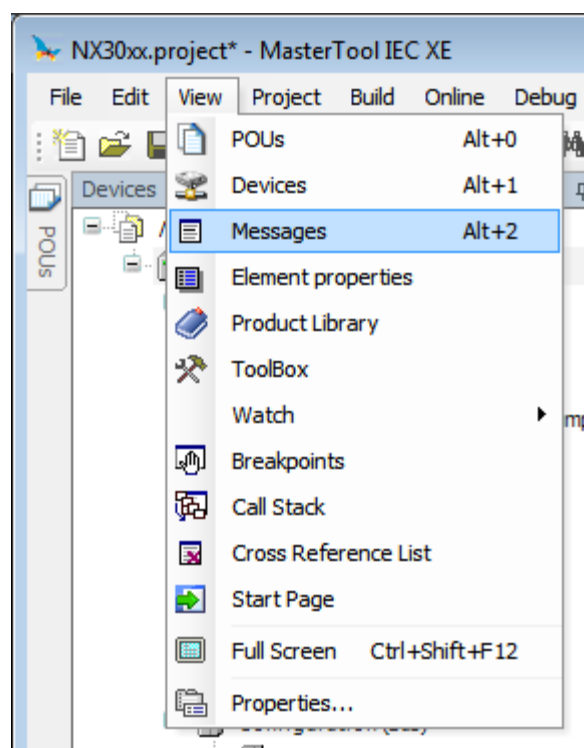


Figure 4-24. Including the Messages on the Screen

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 4-25. After that a warning is displayed in the bottom bar of MasterTool IEC XE that indicates that the tool is operating in Simulation Mode.

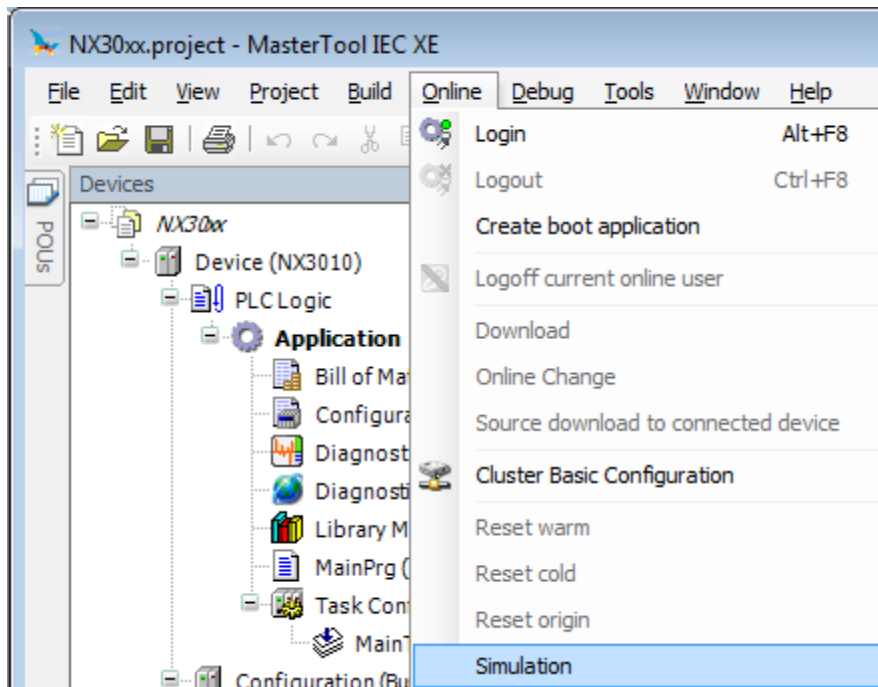


Figure 4-25. Simulation Mode

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.

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

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

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 projectDeclare 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 ApplicationConfigure a Communication Channel to Nexto
- Compile and Load Application on Nexto
- Starting the ApplicationMonitoring the ApplicationDebug an applicationBreakpoint 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 POU's 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 POU's 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 4-26).

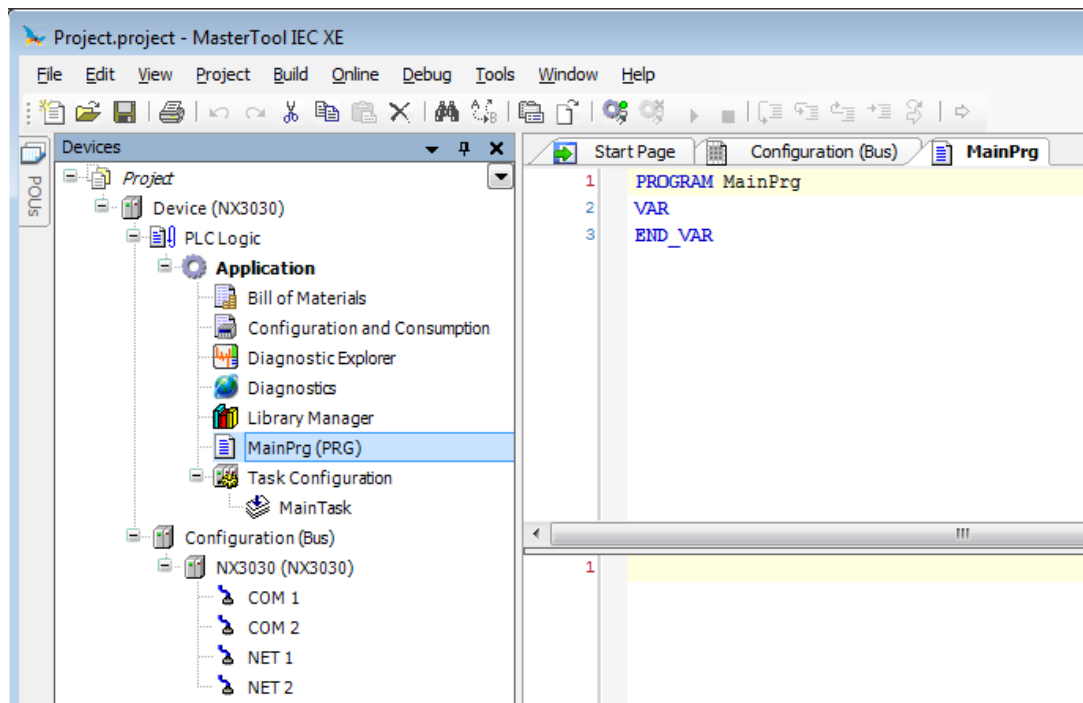


Figure 4-26. ST Editor Window

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.

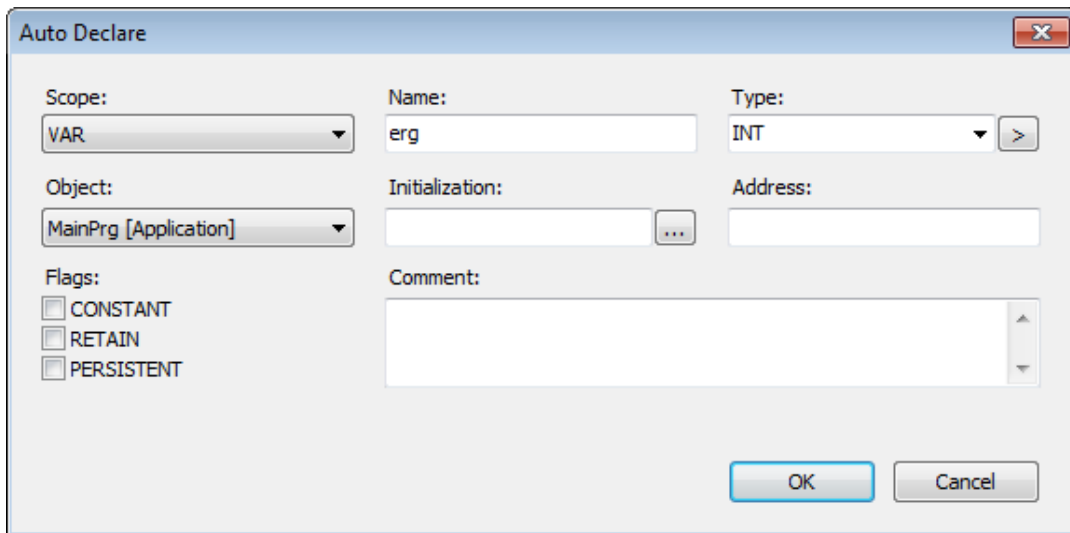


Figure 4-27. Auto Declare Dialog

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 4-28).

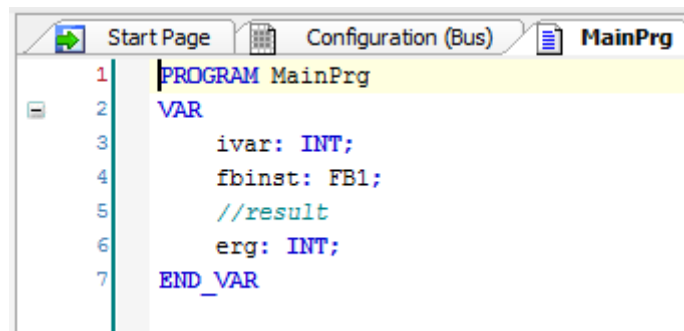


Figure 4-28. “erg” Variable Declaration

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's

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.

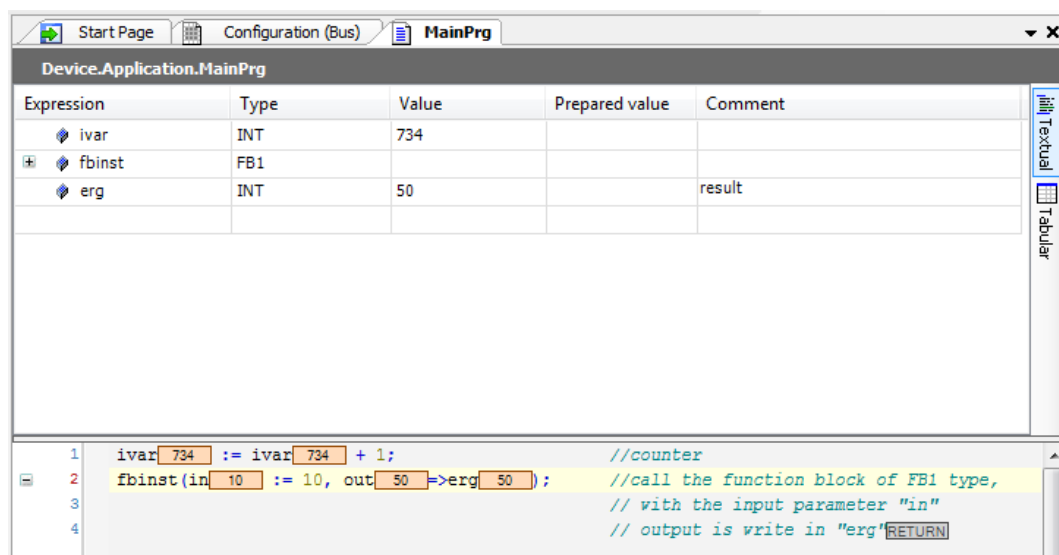


Figure 4-29. Monitoring Expressions and Code Lines

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 4-30. 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 4-31. 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.

```

1  ● ivar 8366 := ivar 8366 + 1; //counter
2  fbinst(in 10 := 10, out 62 =>erg 62); //call the function block of FB1 type,
3  // with the input parameter "in"
4  // output is write in "erg" RETURN

```

Figure 4-32. Application in “Stop” State

A running application will stop at a breakpoint:

```

1  ● ivar 8367 := ivar 8367 + 1; //counter
2  fbinst(in 10 := 10, out 62 =>erg 62); //call the function block of FB1 type,
3  // with the input parameter "in"
4  // output is write in "erg" RETURN

```

Figure 4-33. 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 Nextto 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.

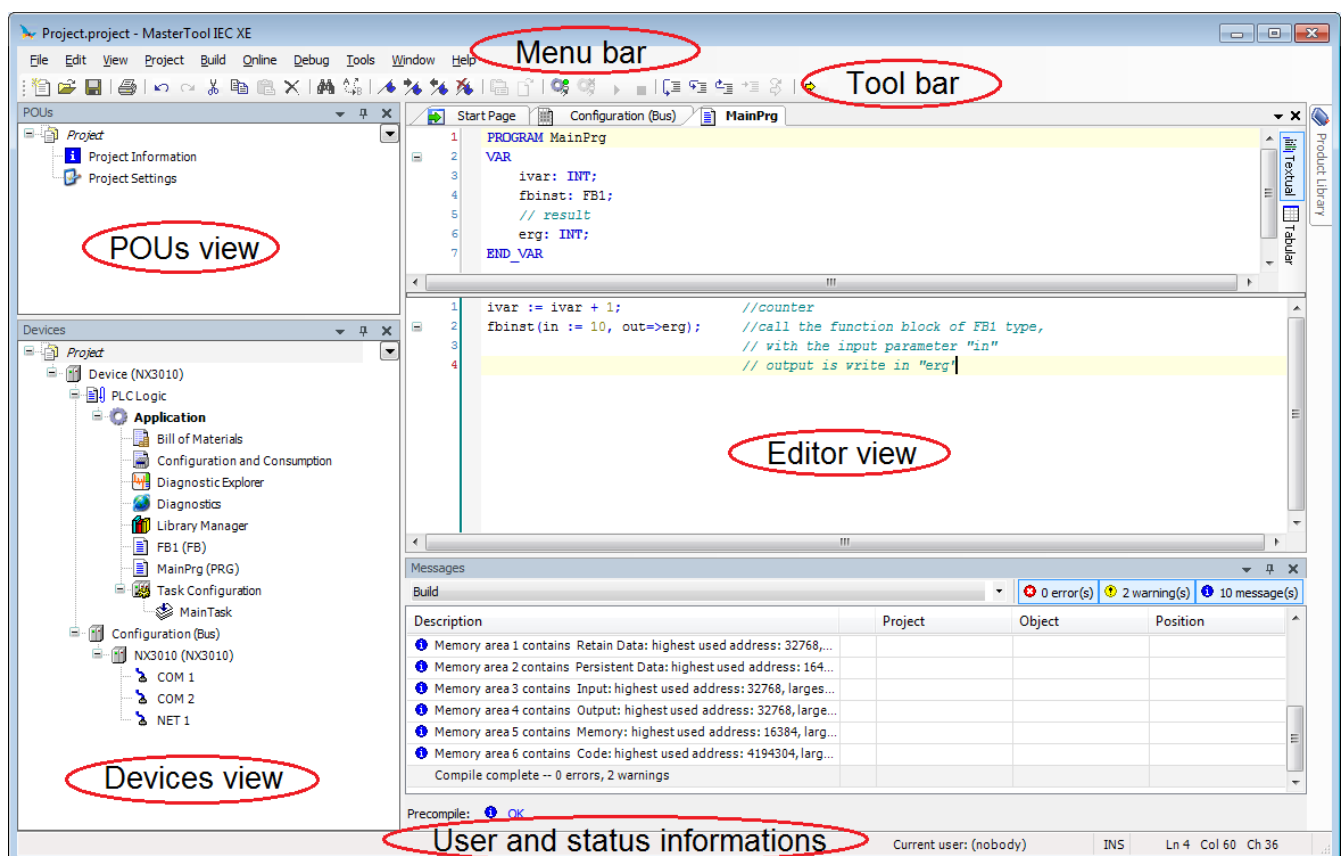


Figure 5-1. Example of the MasterTool IEC XE User Interface

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:
 - **RUN** = program running.
 - **STOP** = program stopped.
 - **HALT ON BP** = program halted on a breakpoint.
 - Program loaded = program loaded on device.
 - Program unchanged = program on device matches that in the programming system.
 - Program modified (Online Change) = program on device differs from that in the programming system, online change required.
 - Program modified (Full download) = program on device differs from that in the programming system, full download required.

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.

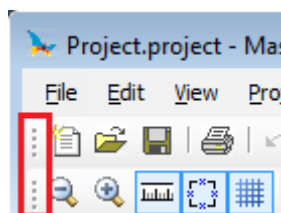


Figure 5-2. Dotted Line on Left Side of Menu and Tool Bars

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.

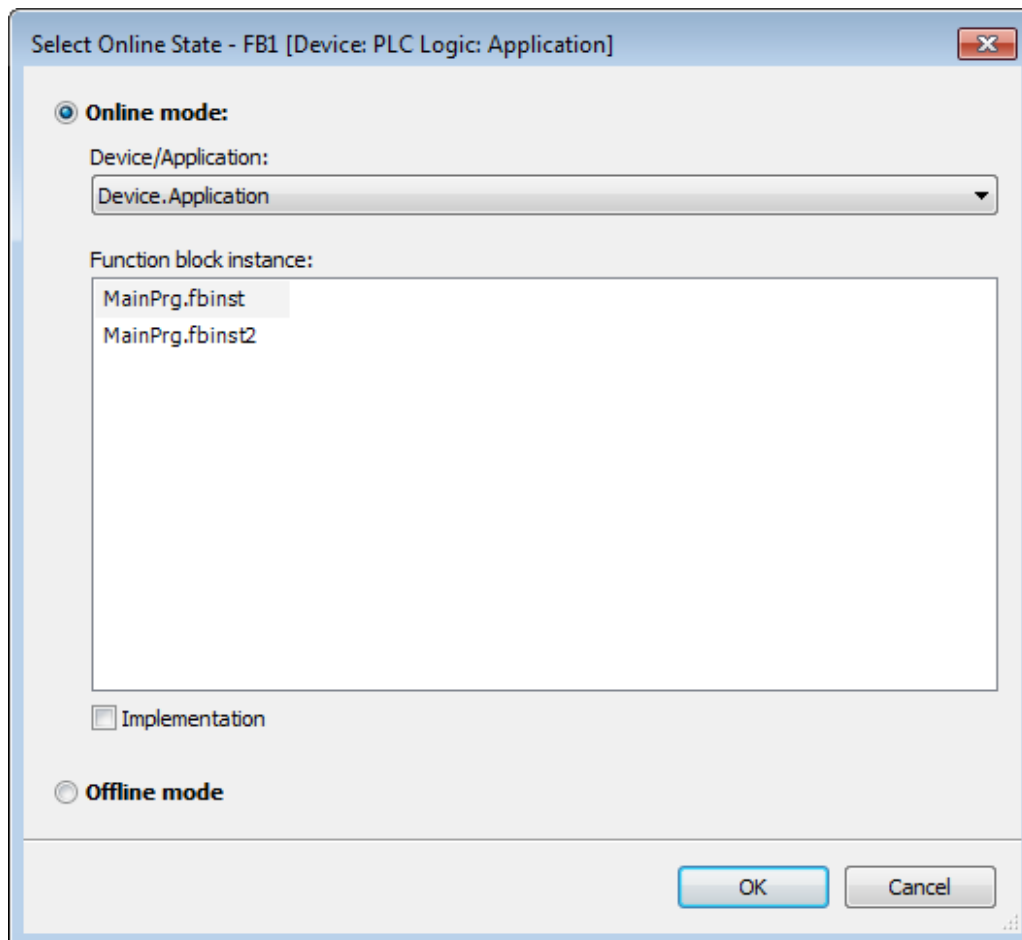


Figure 5-3. Select Online State Dialog

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).

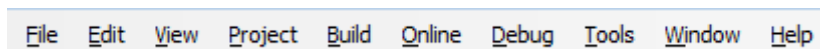


Figure 5-4. Standard Menu Bar

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	
	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	
	Redundancy 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 Watch list	
	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.

This functionality is accessed with a double click over the *Device* item and then selecting the *Files* tab, as showed on Figure 5-5.

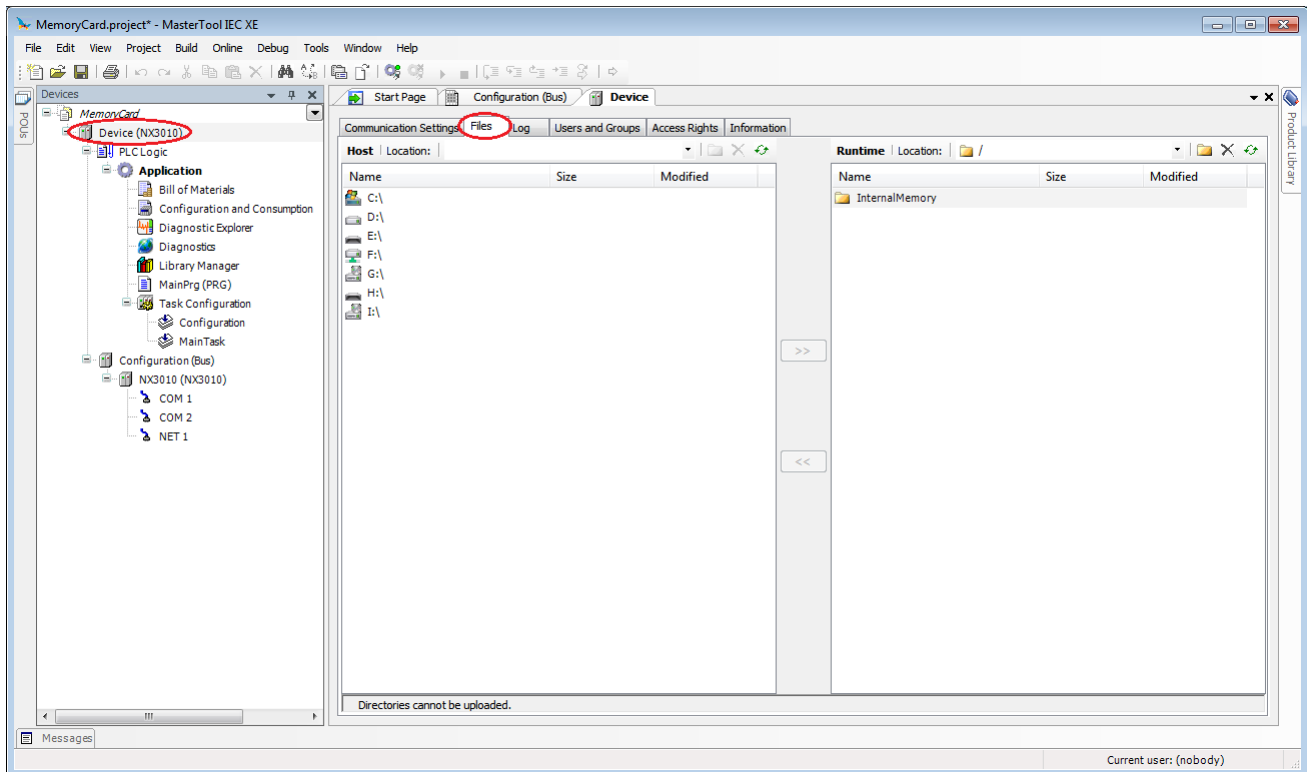


Figure 5-5. User Files Access

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 and Access Right Management of the Project

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:

- | |
|---|
| <ul style="list-style-type: none">- The user passwords are stored irreversibly. If a password gets lost, the respective user account gets unusable. If the Owner password gets lost, the entire project might get unusable.- By default, in new projects, the user <i>Owner</i>'s password is empty. |
|---|

Users

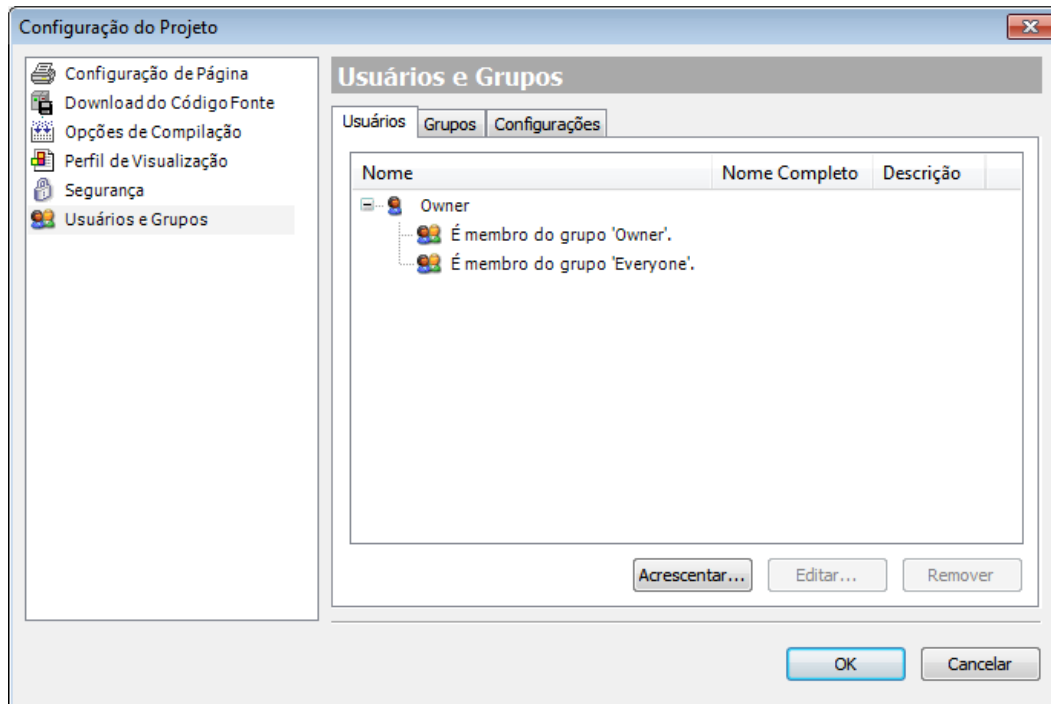


Figure 6-1. Project Settings, Users dialog

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.

Figure 6-2. 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

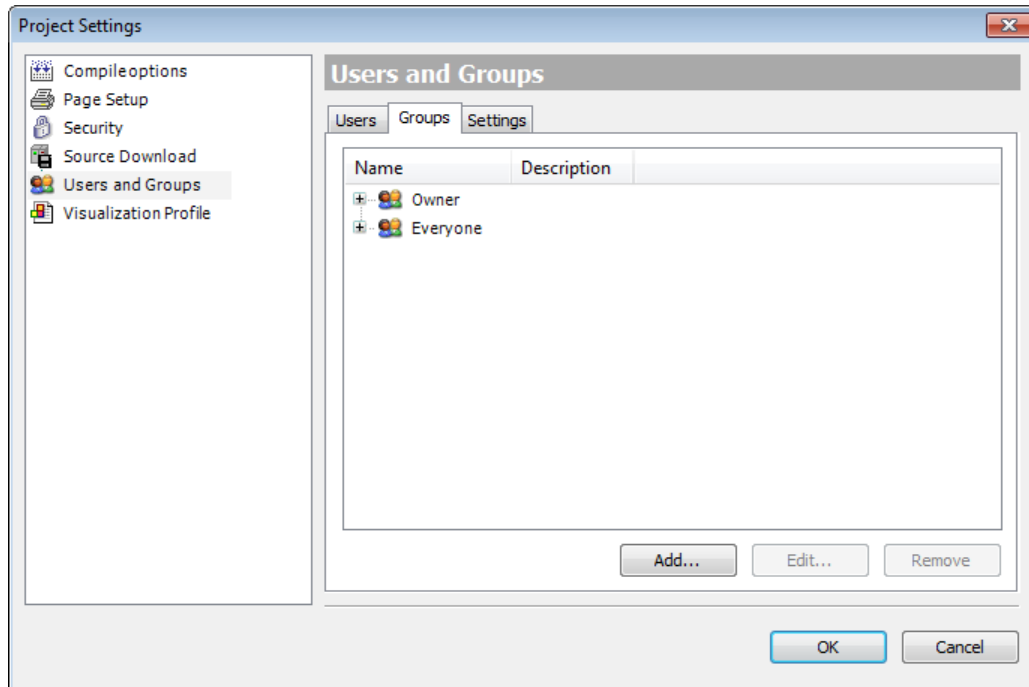


Figure 6-3. Project Settings, Groups Dialog

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.

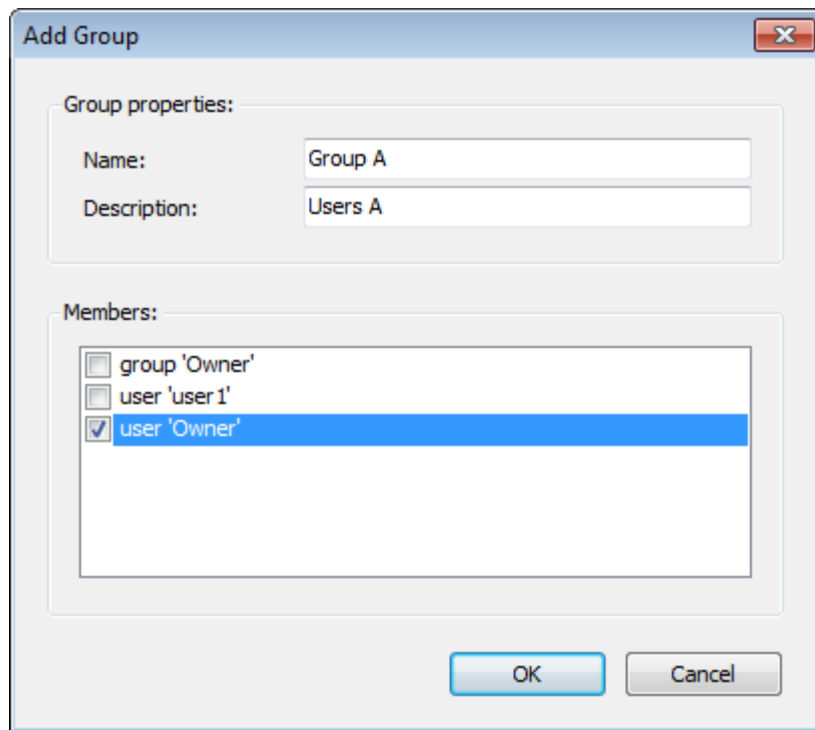


Figure 6-4. Add Group Dialog

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 6-4). 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

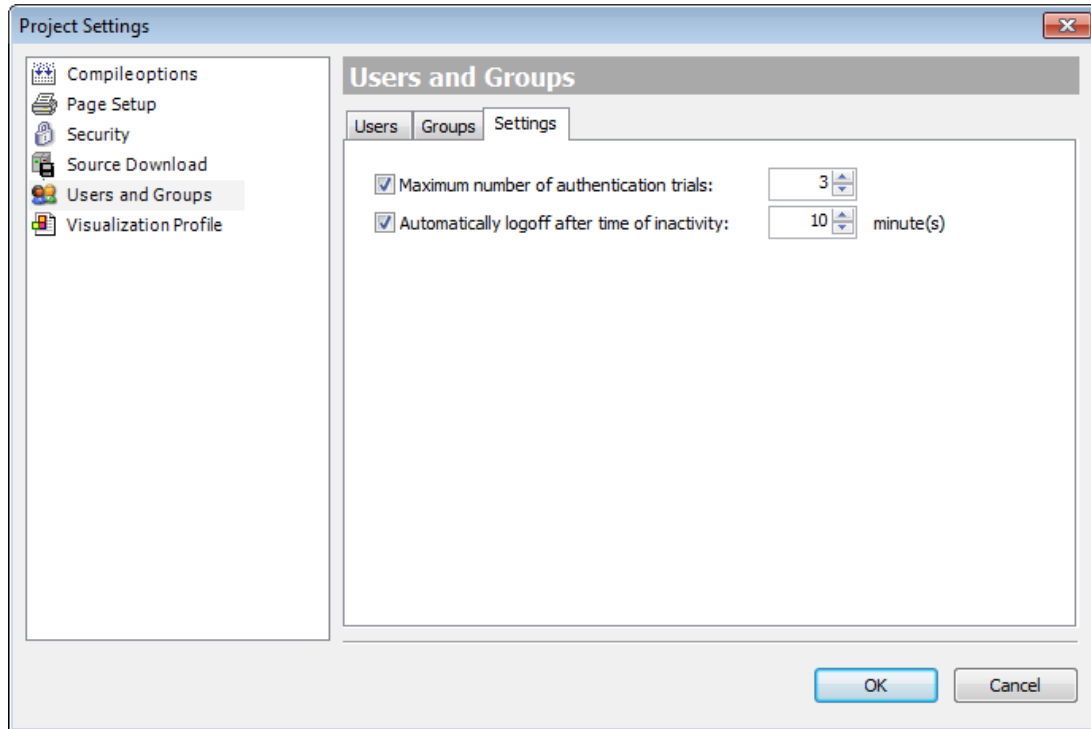


Figure 6-5. Project Settings, Users and Groups Settings Dialog

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.

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”.

To access the *Permissions* screen, click on the option Project > User Management. Then the window in Figure 6-6 will open.

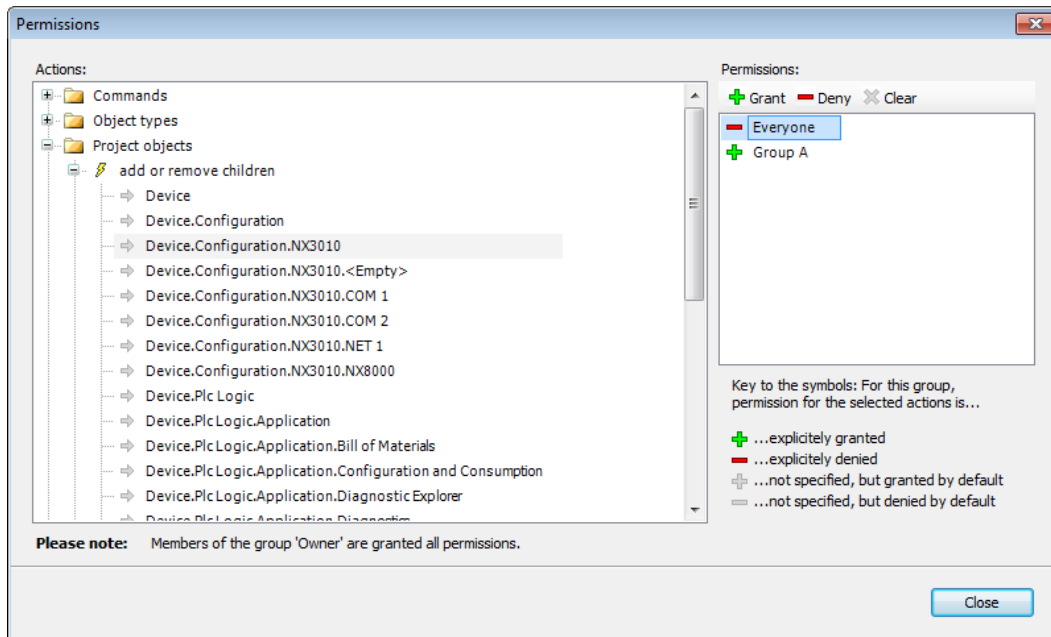


Figure 6-6. Permissions

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.

X 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)

User and Access Right Management of the CPU

Nexto CPUs have a user permissions management system that blocks, or allows, certain actions for each user group in the CPU. To edit these rights in the CPU, the user must access a project in MasterTool IEC XE, with no need to be logged in the CPU. It must then click in the Device Tree, located to the left of the program, double-click the *Device* item, and then select the CPU in the *Communication Settings* tab that will open. Only the *Users and Groups* and *Access Rights* tabs relates with this topic. Figure 6-7 exemplify the steps to access this CPU tab.

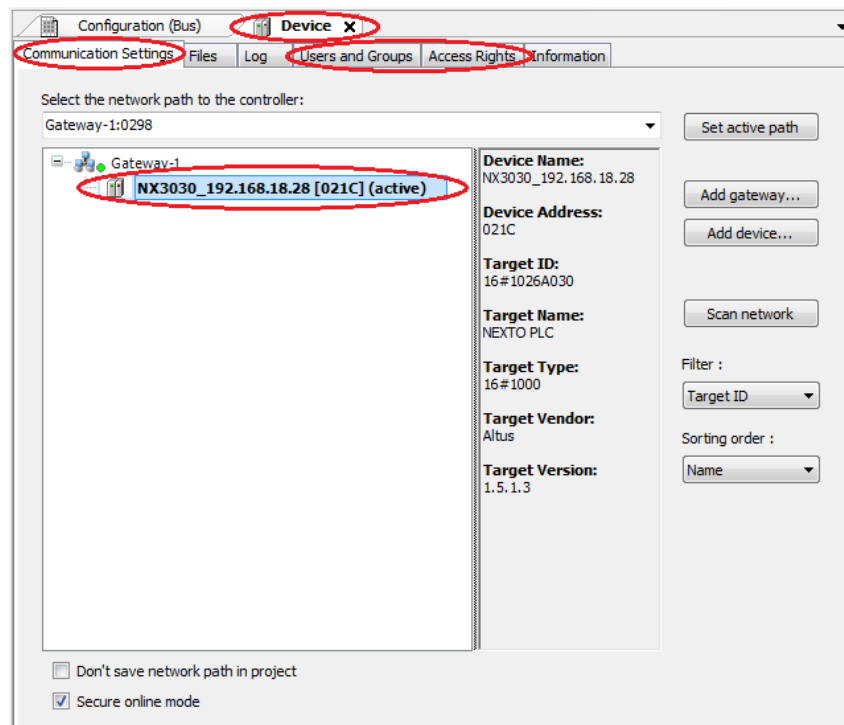


Figure 6-7. Access to Users and Groups and Access Rights tabs

ATTENTION:

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

ATTENTION:

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

Users and Groups

The *Users and Group* dialog is in a tab of the *Device* dialog. It allows the configuration of users and groups accounts that, along with the access rights managements, controls the access to the objects in 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 online user management dialogs is very similar to that of the project's user management. There is even the possibility to "import" user account definitions from the project's user management.

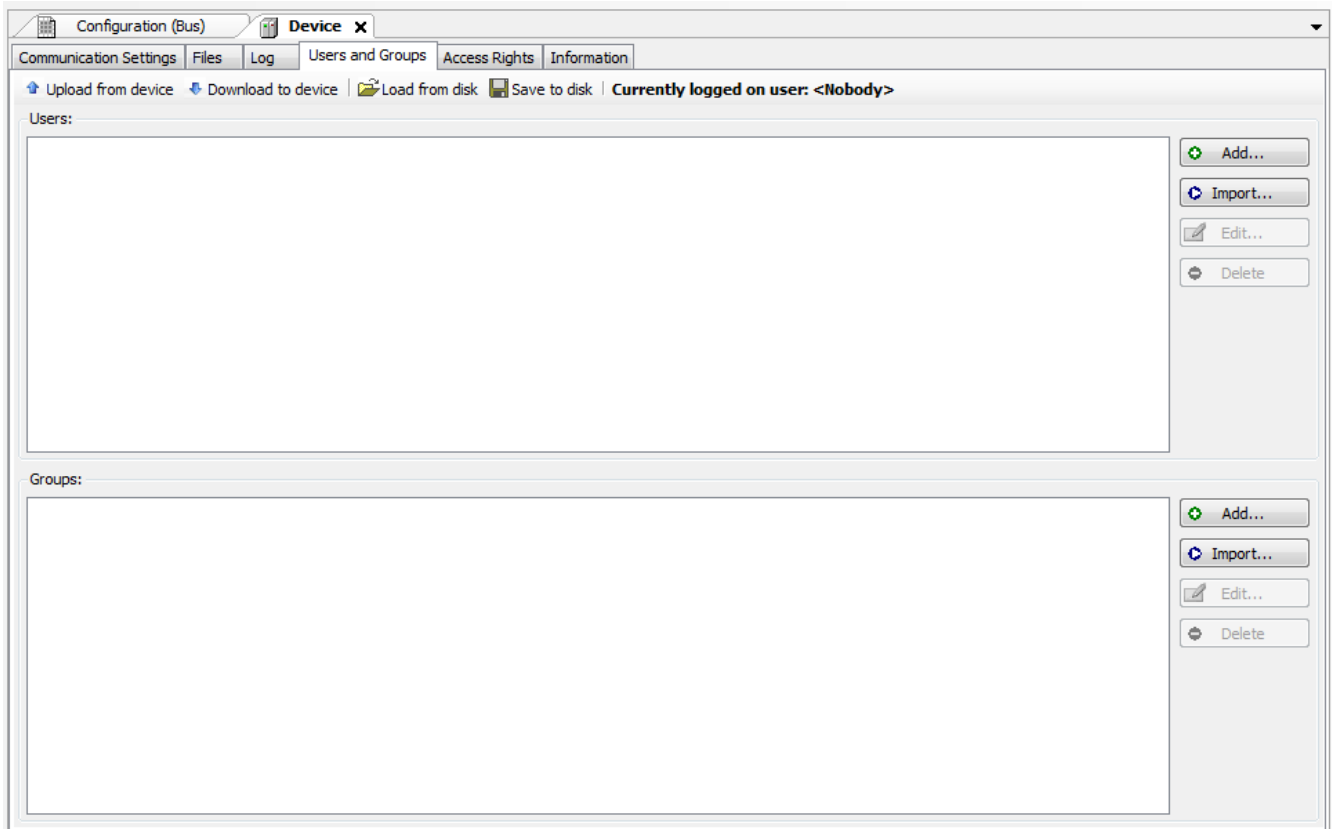
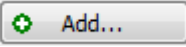


Figure 6-8. Device Dialog, Users and Groups

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.

ATTENTION:

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

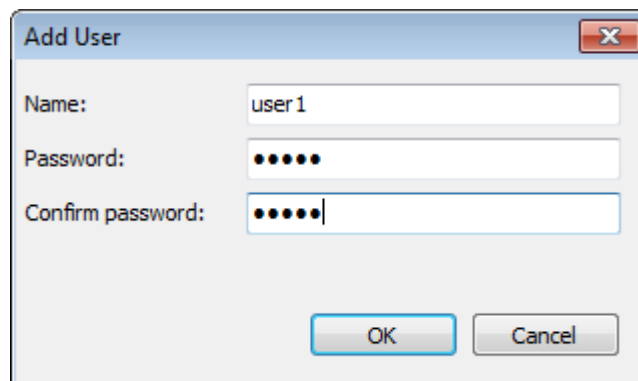

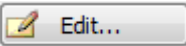


Figure 6-9. Add User Dialog

: 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.

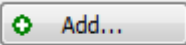
ATTENTION:

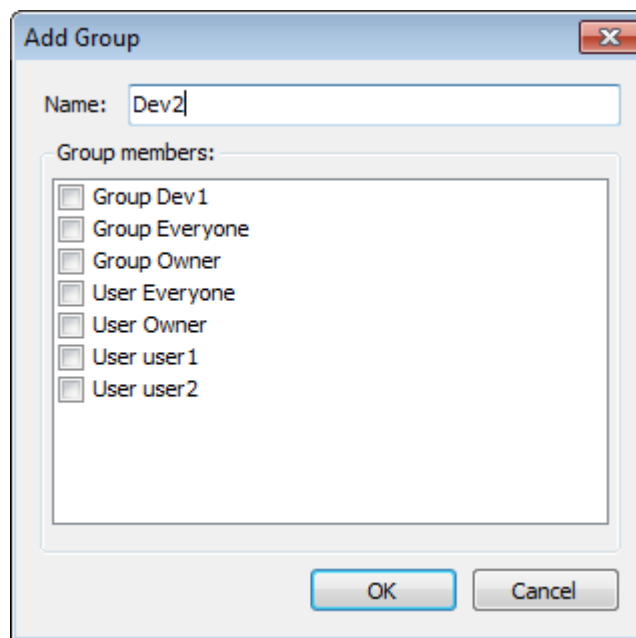
The users and groups of the CPU can import the users and groups of the project, however they won't have the same organization, because the project access rights are different from the CPU access rights, so the users and groups structure will have to be organized again.


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

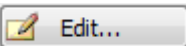
: The currently selected user account will be deleted.


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 6-10. 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.

 Upload from device ,  Download to device : 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).

Considerations about Default Users and Groups

The default users and groups in the device and loaded into the project through the Upload from device button were changed between Nexto CPUs firmware versions.

In firmware versions 1.3.x.x or lower, the existing users and groups are: Everyone and Owner (Table 6-1).

Users	Groups
Everyone	Everyone
Owner	Owner

Table 6-1. Users and Groups in Versions 1.3.x.x

However, in firmware versions 1.4.x.x or higher there are the users: Administrator and Everyone, and the groups: Administrator, Developer, Everyone, Service e Watch (Table 6-2).

Users	Groups
Administrator	Administrator
Everyone	Developer
	Everyone
	Service
	Watch

Table 6-2. Users and Groups in Versions 1.4.x.x

For Firmware Versions 1.3.x.x or Lower

For firmware versions 1.3.x.x or lower, the following groups and users are defined by default in the Nexto Series CPUs.

Group Everyone

This is the default group to perform accesses in a CPU while there are no defined users and groups.

Group Owner

This group have privileges and it's not possible to remove it in firmware versions 1.3.x.x or lower.

User Everyone

The user Everyone is defined in the group Everyone. This user doesn't have a defined password.

User Owner

The user Owner is in the group Owner. The default password for this user is "Owner" and it can be modified.

For Firmware Versions 1.4.x.x or Higher

For firmware versions 1.4.x.x or higher, the following users and groups are defined as default in the Nexto Series CPUs. This division into a higher number of groups aim at presenting an initial proposal of different users levels that can access the CPU.

Group Administrator

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

Group Developer

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

Group Everyone

This is the default group to perform accesses to a CPU while there are no defined users and groups.

Group Service

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

Group Watch

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

User Administrator

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

User Everyone

The user Everyone is defined in the groups Everyone and Administrator. This user doesn't have a defined password.

Users and Groups from Old Projects

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

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

ATTENTION:

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

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 6-11 shows the permission to add and remove children to/from the PLC Logic object granted for user groups *Everyone* and *Owner*.

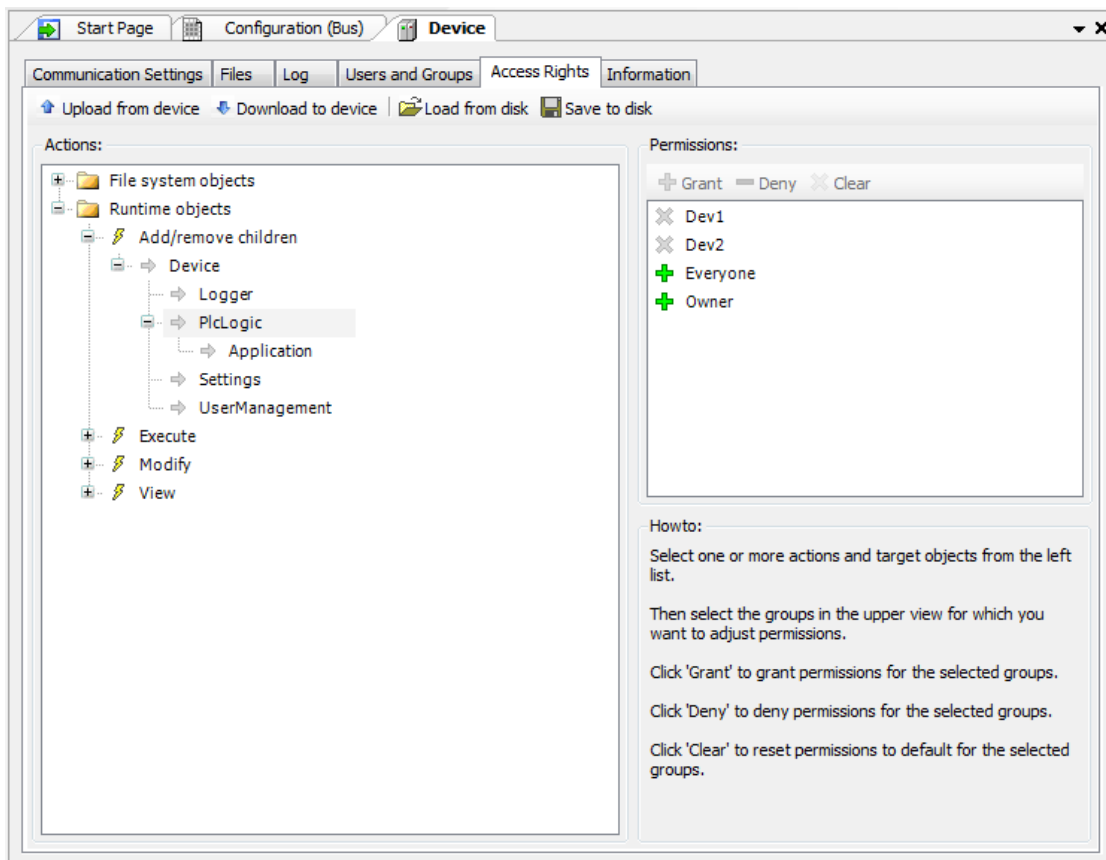


Figure 6-11. Device, Access Rights

The access rights (add / remove children, execute, modify, view) are configured to each device and enable for each user the actions described in Table 6-3.

Devices		Action	Relevant rights				
			Add / Remove Children	Execute	Modify	View	
Device	Logger		Login to device			x	
			Read Log Inputs			x	
	PlcLogic	Application	Login to application			x	
			Download of an application into a CPU without application	x			
			Download of an application / Online change			x	
			Create boot application			x	
			View variables			x	
			Write values in variables			x	x
			Force values in variables			x	x
			Add/Remove Breakpoint			x	
			Step Into/Over/Out		x		
			Run to Cursor		x		
			Set Next Statement		x		
			Read Call Stack				x
			Start/Stop application		x		
			Perform Reset Warm/Cold/Origin			x	
			UserManagement		View user configuration		
	Modify user configuration					x	

Table 6-3. Actions and 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.

ATTENTION:


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

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.

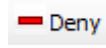
: The action(s) currently selected in the *Actions* window are denied for the group.

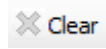
: Currently there is no explicit access right definition for the action(s) currently selected in the *Actions* window.

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:

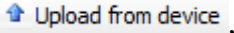
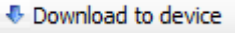
 **Grant**: Explicit granting access permission.

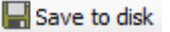
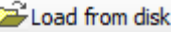
 **Deny**: Explicit denying access permission.

 **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).

Access Rights of Old Projects

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

In the new Nexto CPU or in a updated CPU, run the command *Load from disk*, then select the file generated before, and finally run the command *Download to Device*, thus sending the configuration to the CPU.

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*.

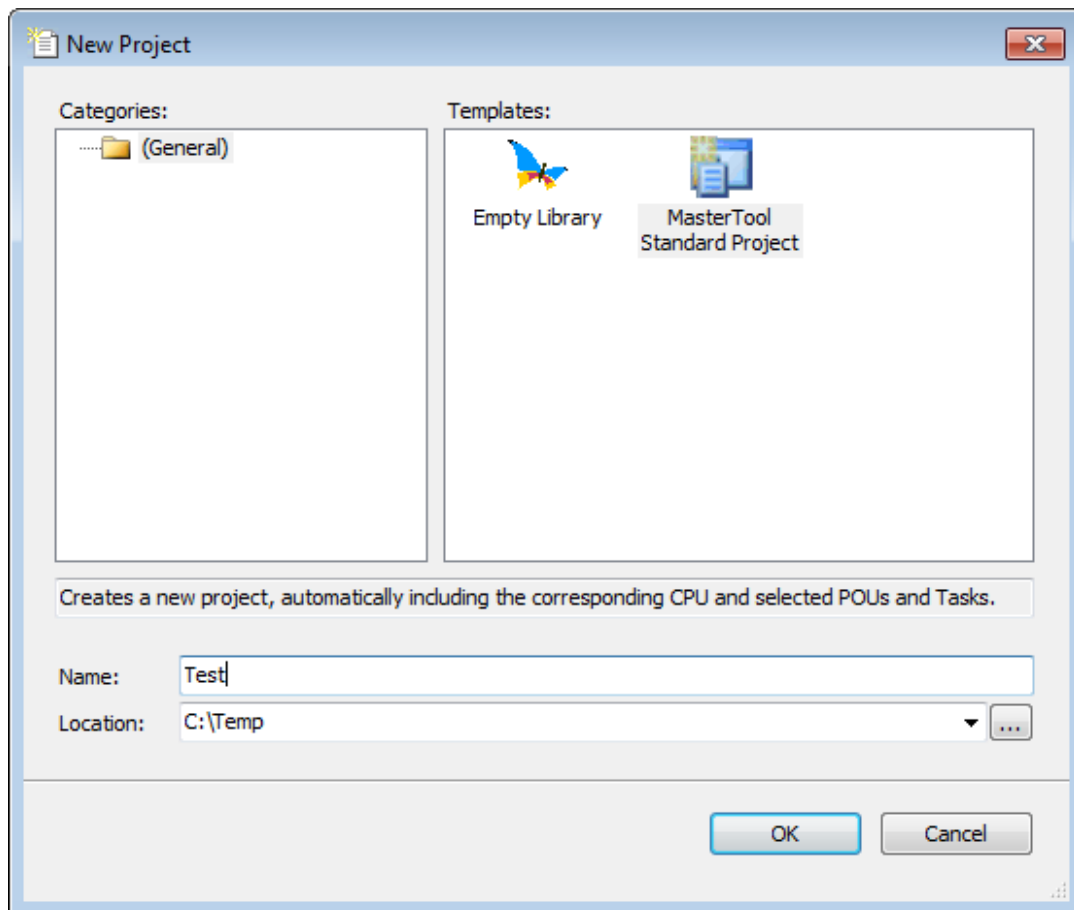




Figure 7-1. New Project

- *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, POUs 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.

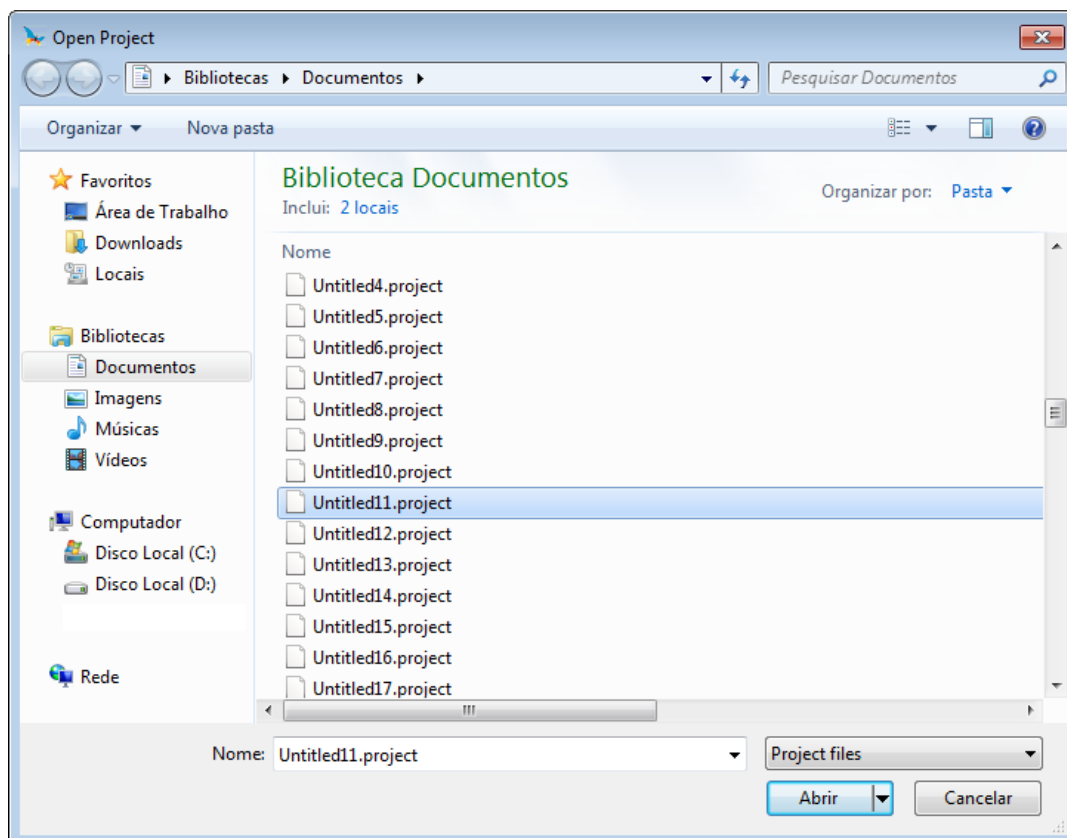


Figure 7-2. Open Project

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.

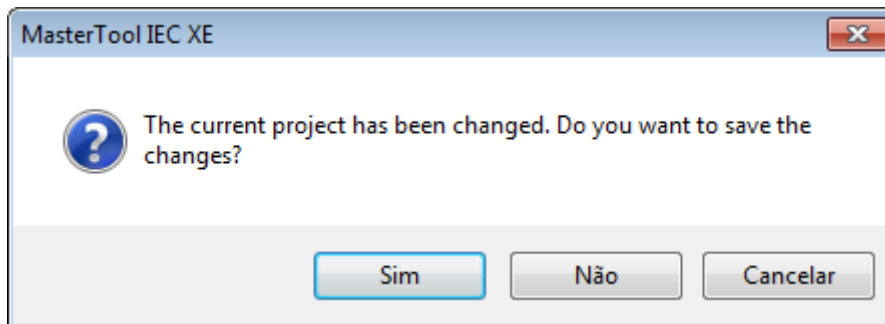


Figure 7-3. Project Alterations

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.

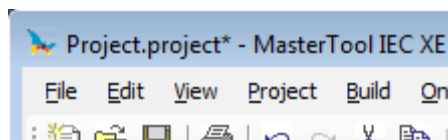


Figure 7-4. Title bar, Modified Project

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

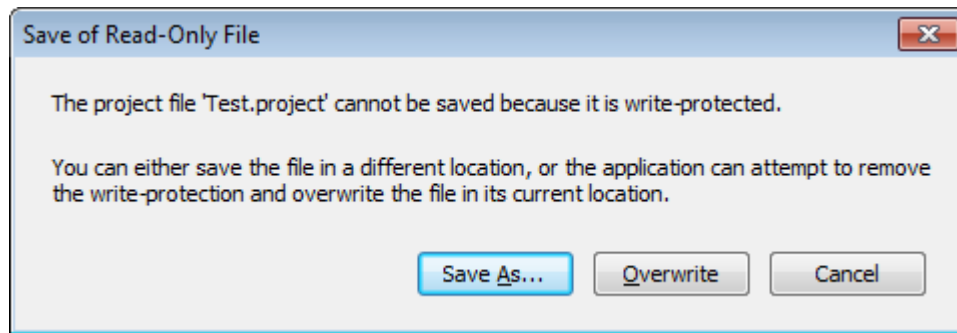


Figure 7-5. Message Dialog, Project Write-Protected

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.

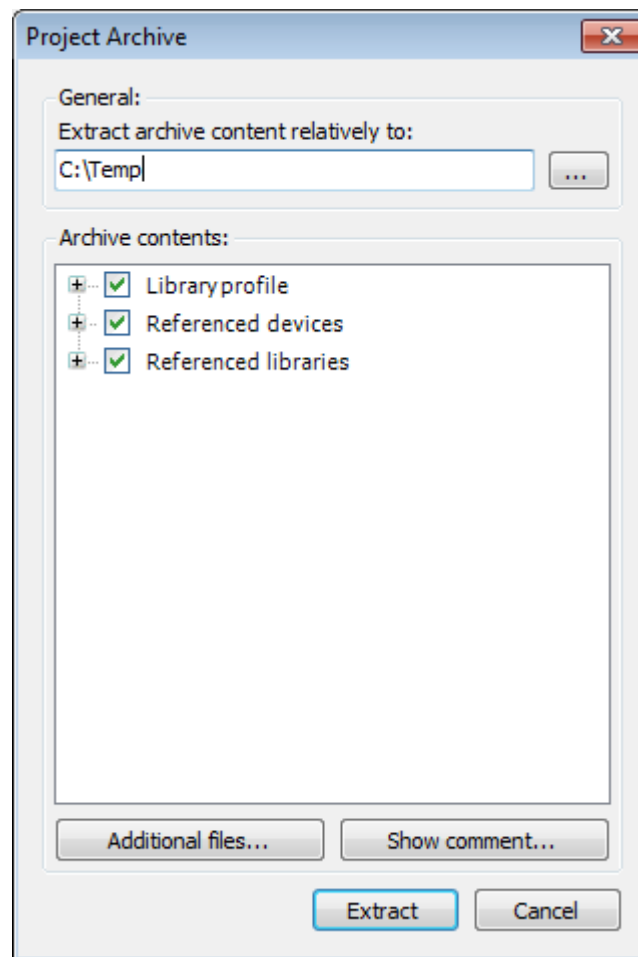







Figure 7-6. Project Archive

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:

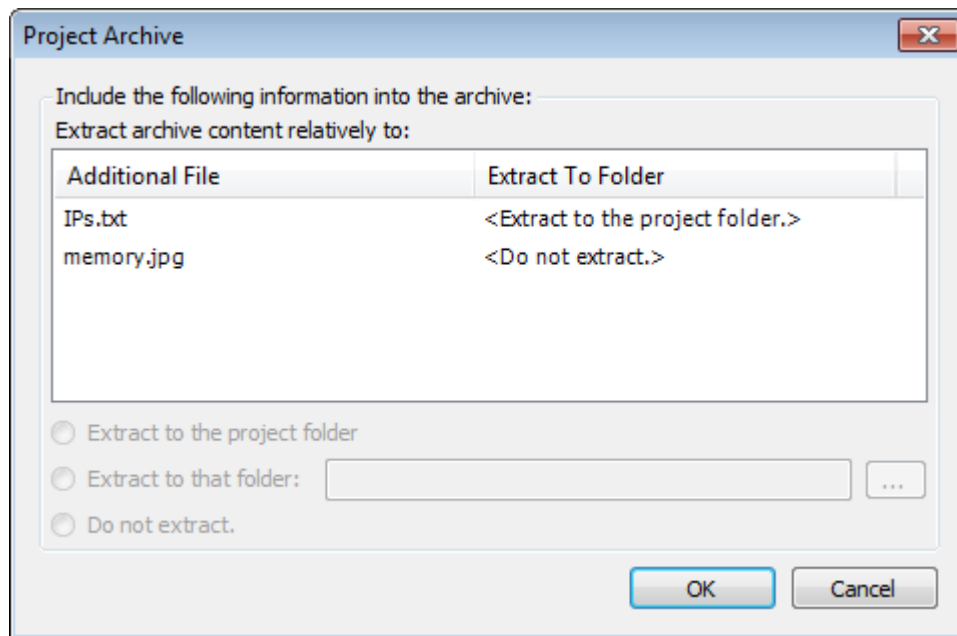



Figure 7-7. Additional Files Dialog

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.

Extracting Files from Projects Created in Older Versions

In many situations, when a project is created in a MasterTool IEC XE version, and it's then opened in a later version, it's necessary to run a Project Update command. This project update, in most cases, will cause the project to be changed and therefore make it no longer possible to execute a login command in the CPU. Probably it won't also be possible to execute an Online Change. To avoid this

kind of situation it's possible to create a Project Archive (*.projectarchive) with the old version of the MasterTool IEC XE. This project file can be opened in the new version of MasterTool IEC XE and so the login can be done.

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

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.

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

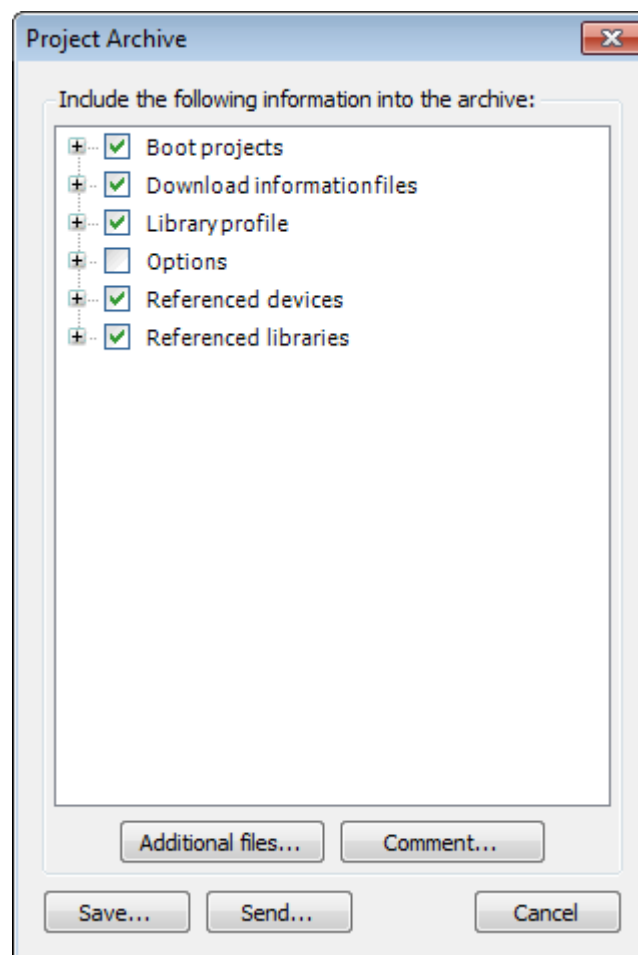



Figure 7-8. Project Archive Dialog

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.

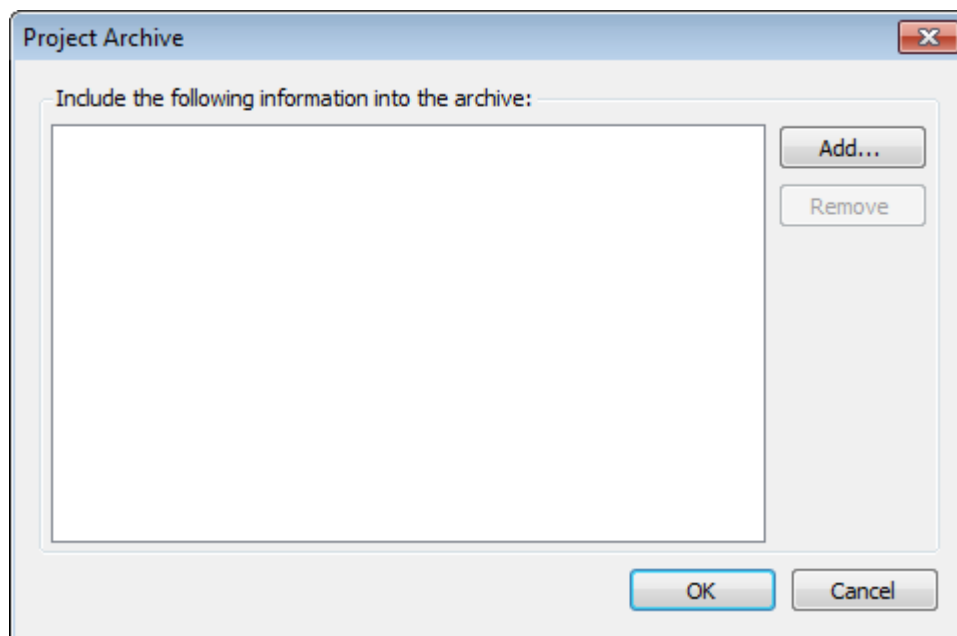


Figure 7-9. Additional Files

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.

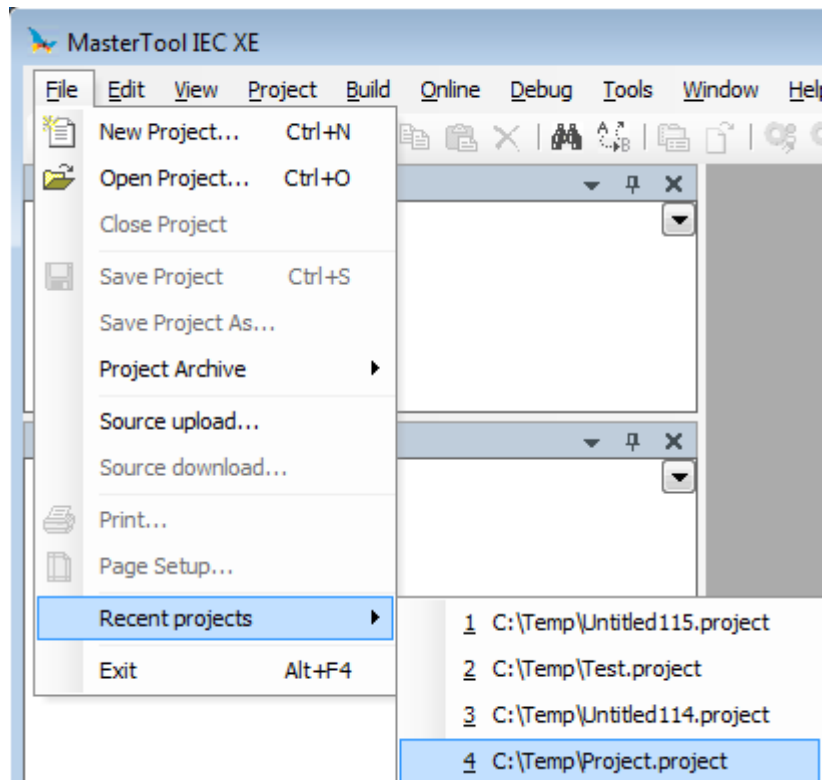


Figure 7-10. Recent Projects List

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:

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.

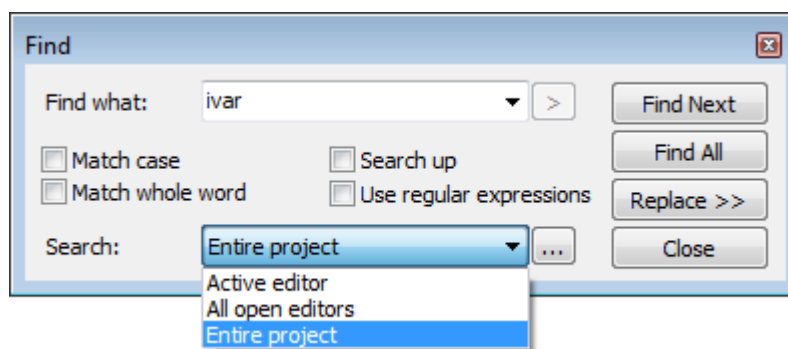






Figure 7-11. Find Dialog

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 .

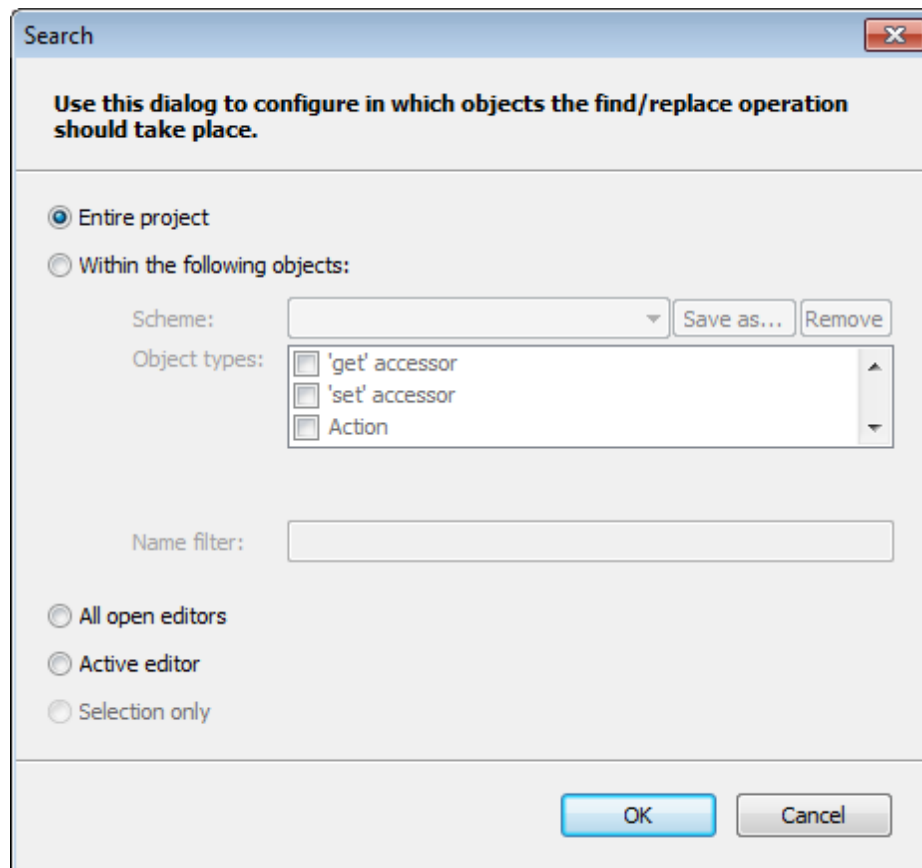



Figure 7-12. Search Dialog

- *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.

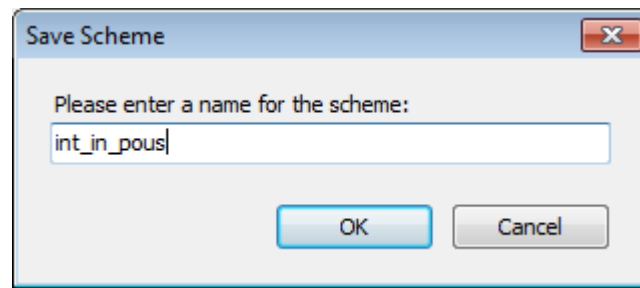


Figure 7-13. Save Scheme Dialog

- *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.

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.

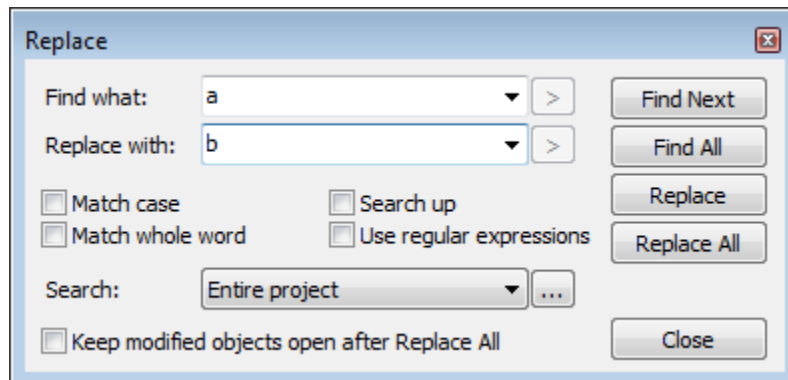


Figure 7-15. Replace

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.

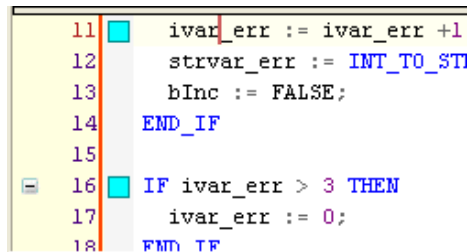


Figure 7-16. Bookmarks in ST Editor

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

In the *Text Search* tab it's possible to search for a specific item (Figure 7-17). By typing one or more characters in the search field, all items that contain the searched text are going to be listed. It's possible to restrain the search to a variable category through the *Filter* field. With a double-click over the item, it's going to be inserted in the current cursor position in the editor.

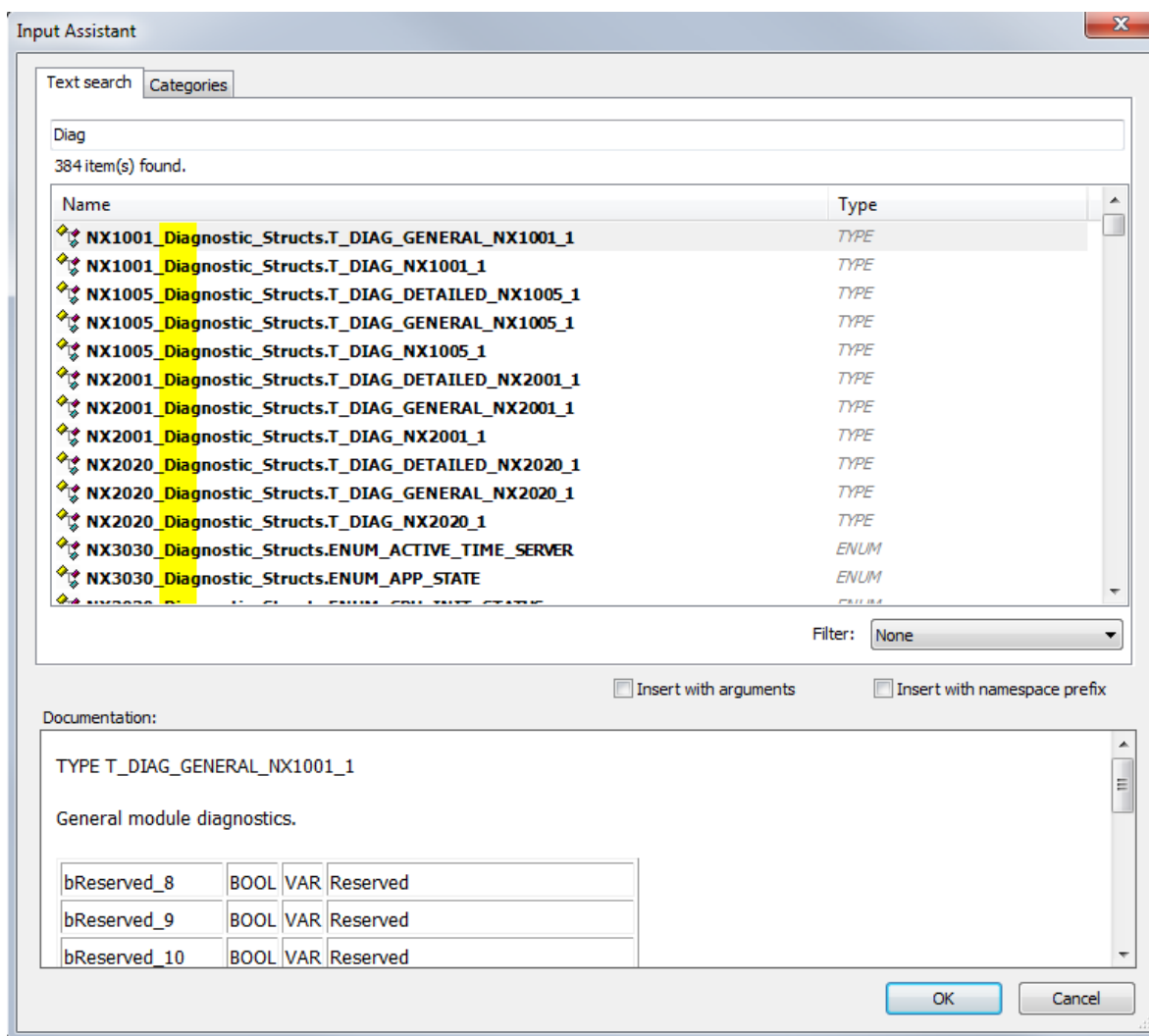


Figure 7-17. Text Search Tab

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

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

NOTES:

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

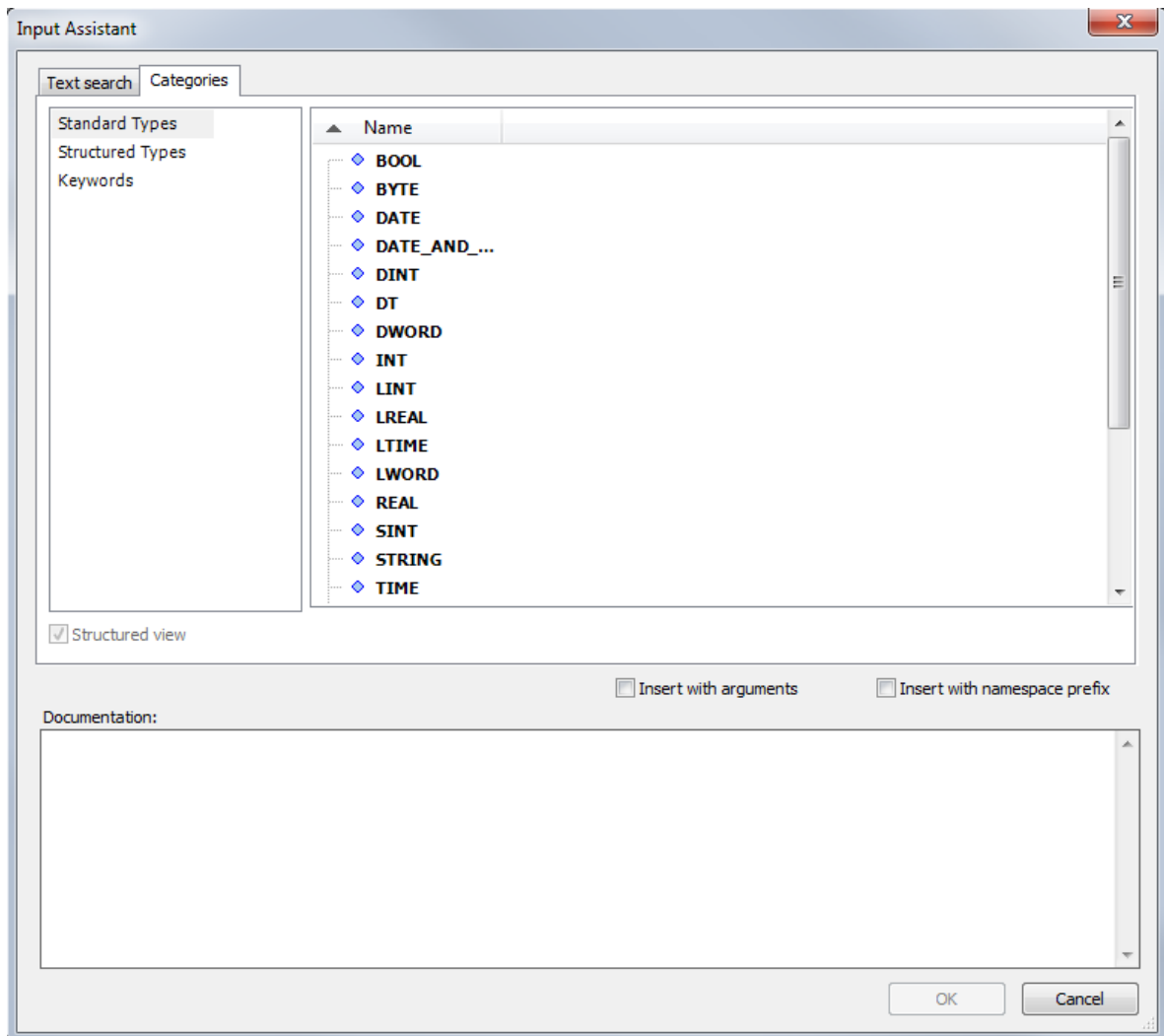


Figure 7-18. Categories Tab

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

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

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

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

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.

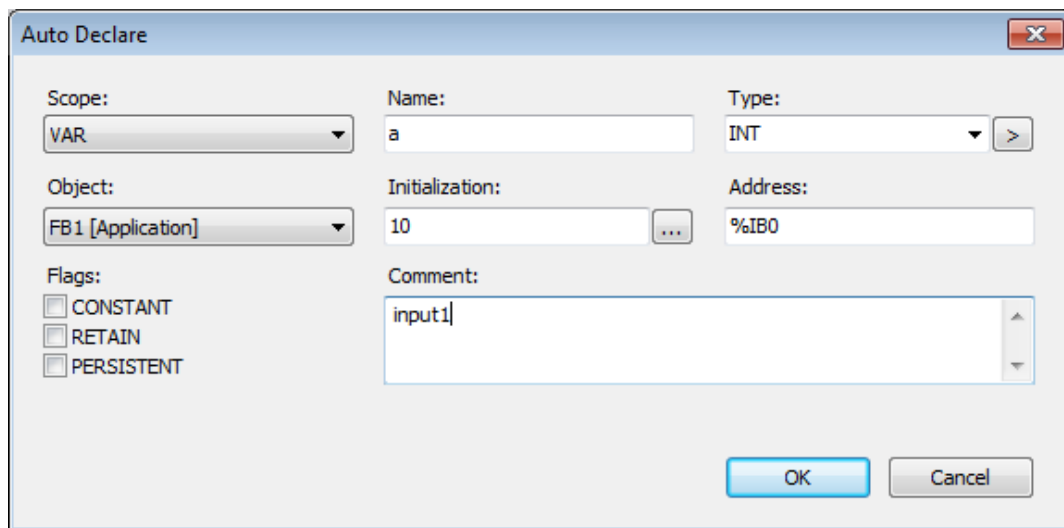


Figure 7-19. Dialog Box for Declaration of Variables


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:

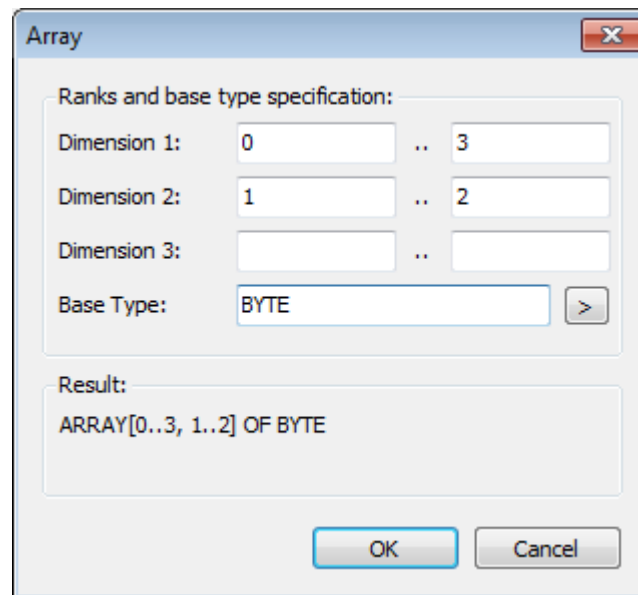



Figure 7-20. ARRAY

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.

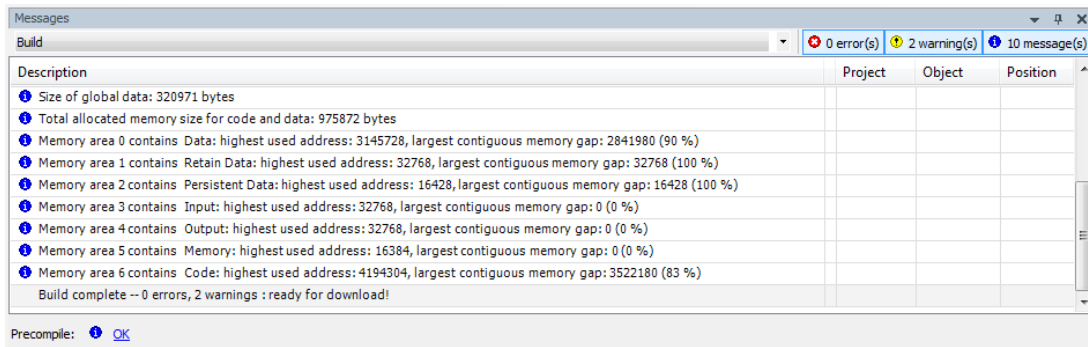


Figure 7-21. Messages Window

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 7-21) 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 Watch list View provides a submenu for opening the available watch lists.

Watch

Symbol: 

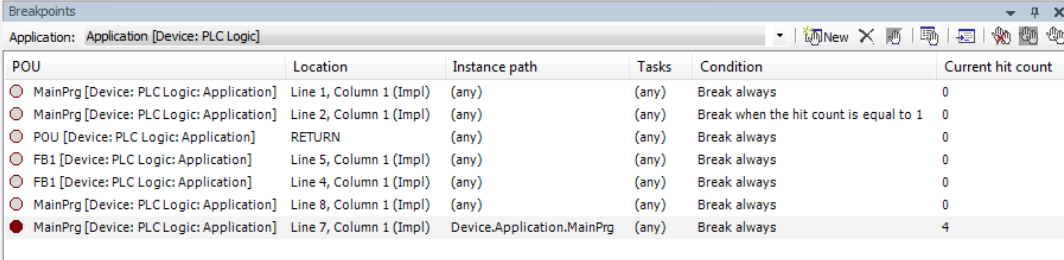
This command opens a submenu with *commands Watch 1, Watch 2, Watch 3, Watch 4 and Watch all Forces*. These are used to open the respective watch list in a view window. *Watch all Forces* is a special watch view for the currently forced values. For further information, see **Debug Menu**.

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 7-22. 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.

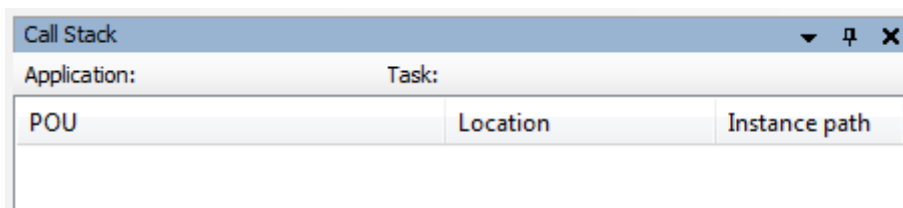
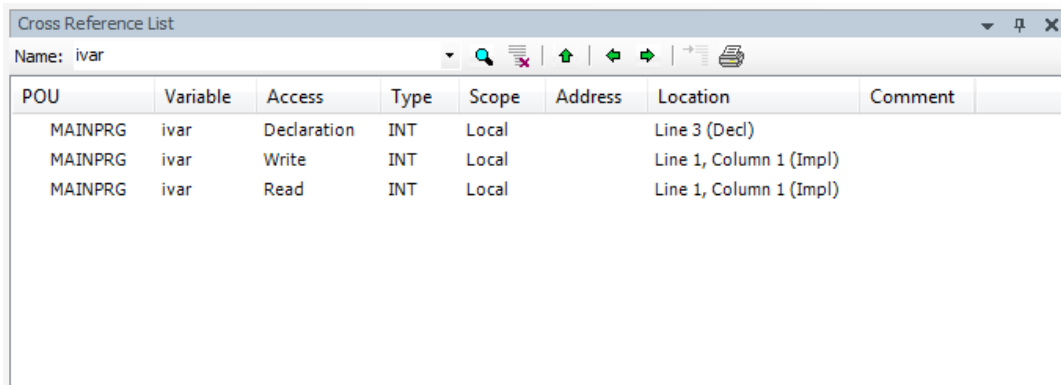


Figure 7-23. Call Stack View


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 (Decl)	
MAINPRG	ivar	Write	INT	Local		Line 1, Column 1 (Impl)	
MAINPRG	ivar	Read	INT	Local		Line 1, Column 1 (Impl)	

Figure 7-24. 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.

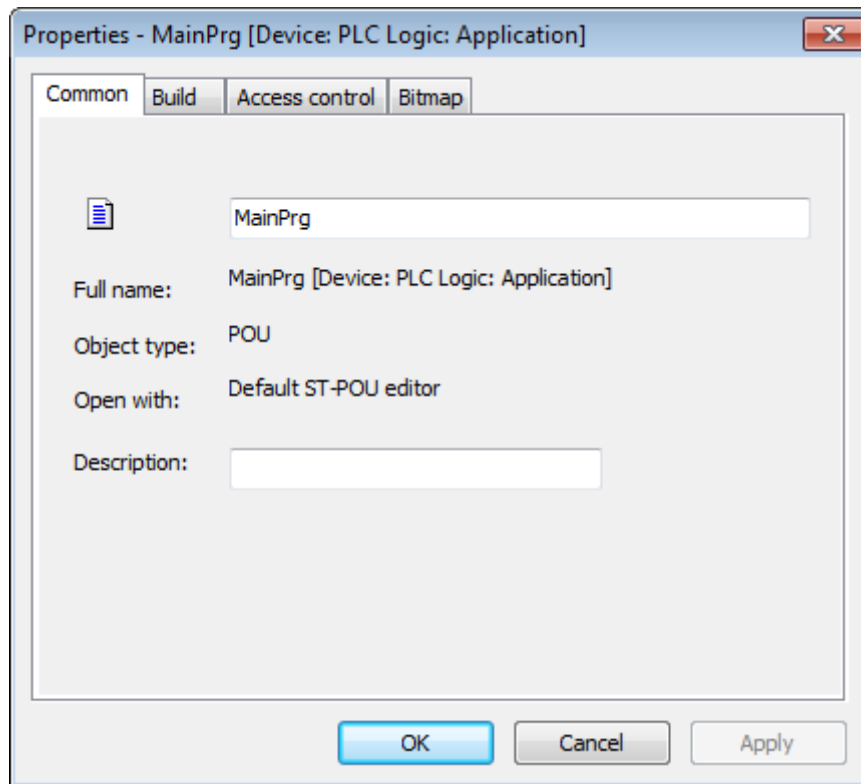


Figure 7-25. Properties Dialog, Common Category

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.

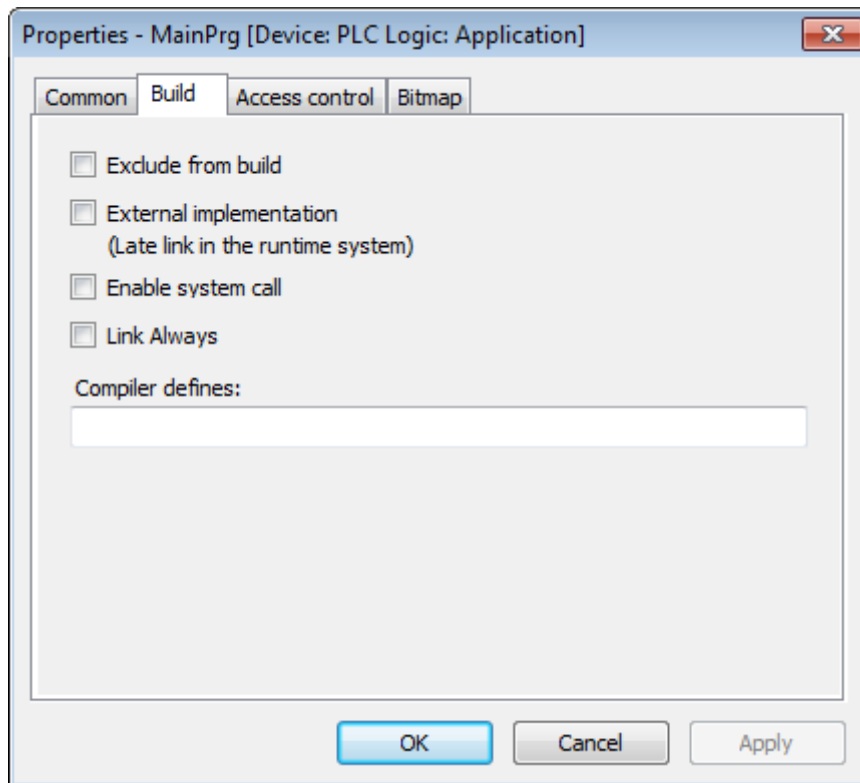


Figure 7-26. Properties Dialog, Category Build

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.

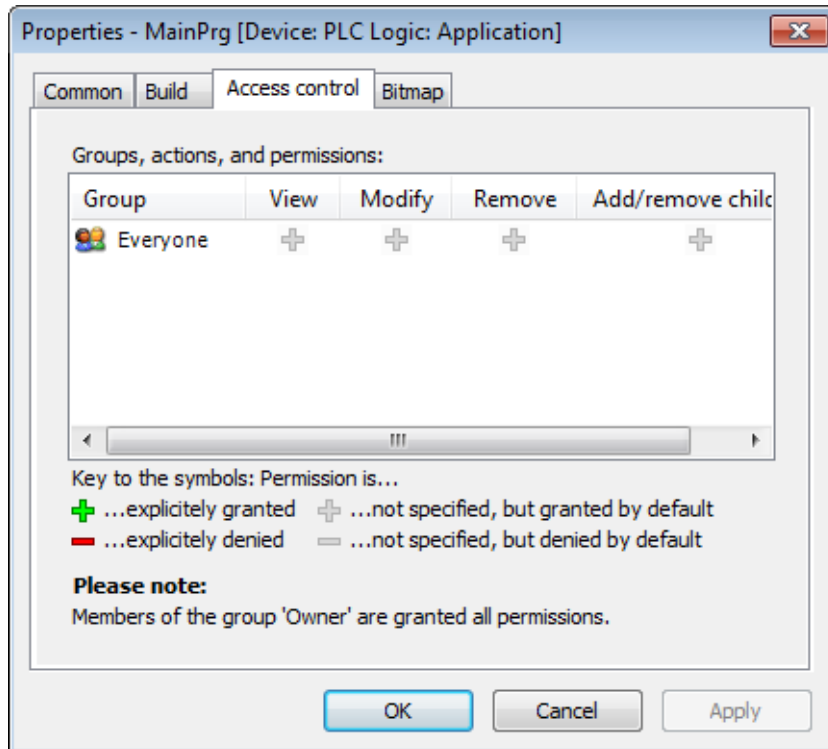


Figure 7-27. Properties Dialog, Category Access Control

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.

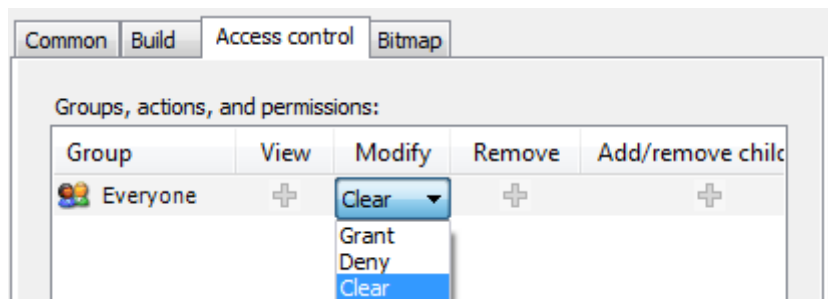


Figure 7-28. Selection List of Rights for action “Modify” for group “Everyone”

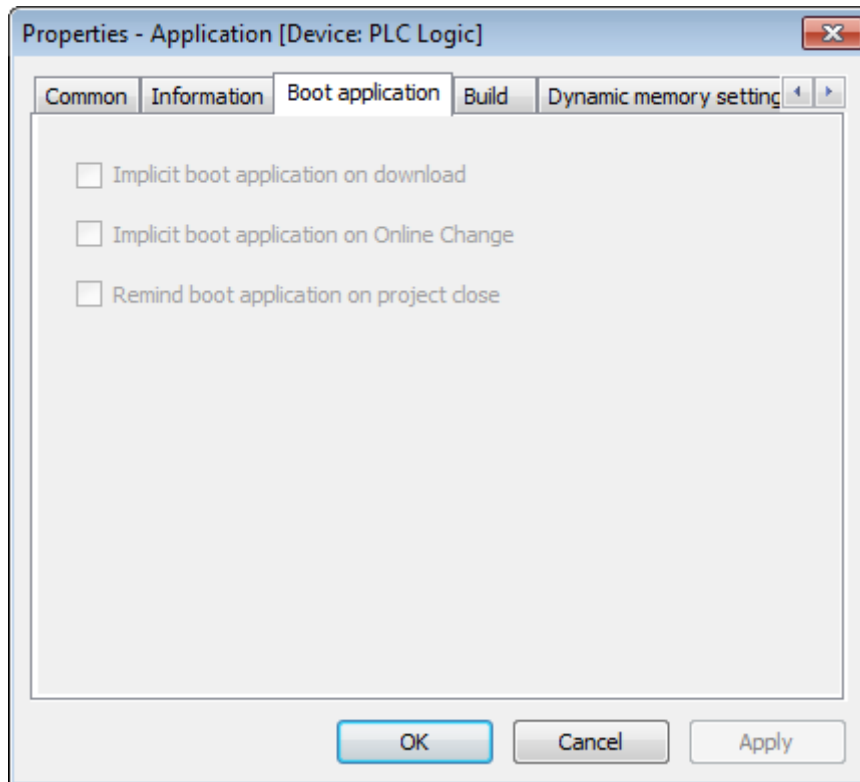
Boot Application

Figure 7-29 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.

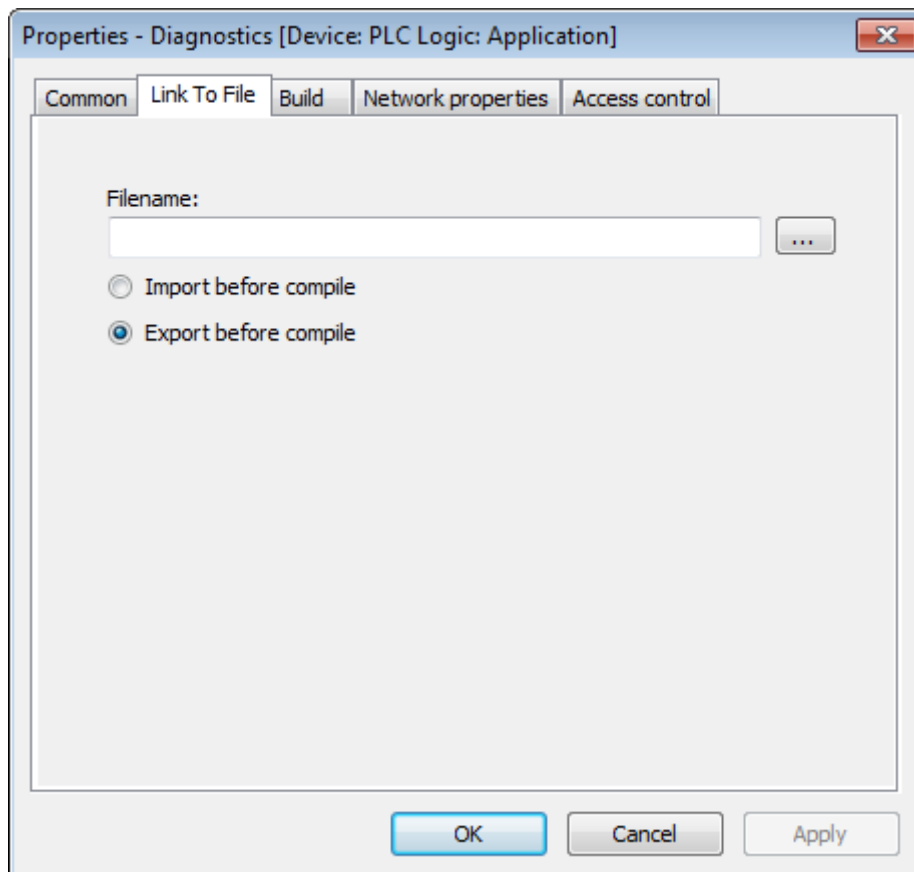


Figure 7-30. Properties Dialog, Category Network Settings

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.

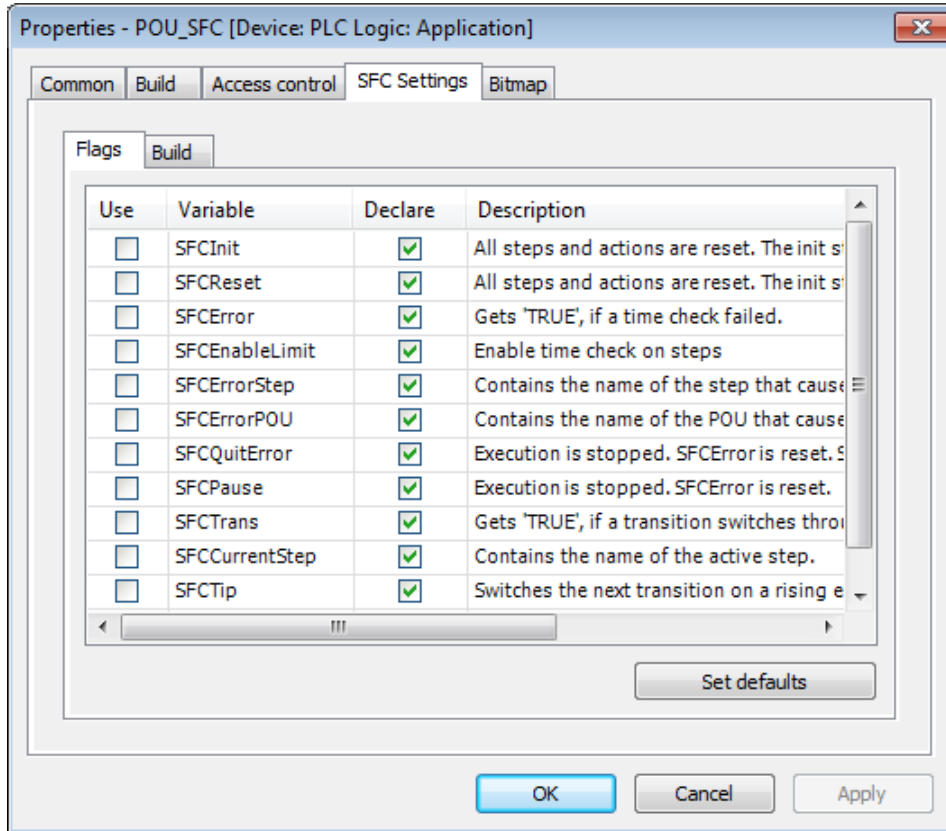


Figure 7-31. Properties Dialog, Category SFC Settings, Flags

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**.

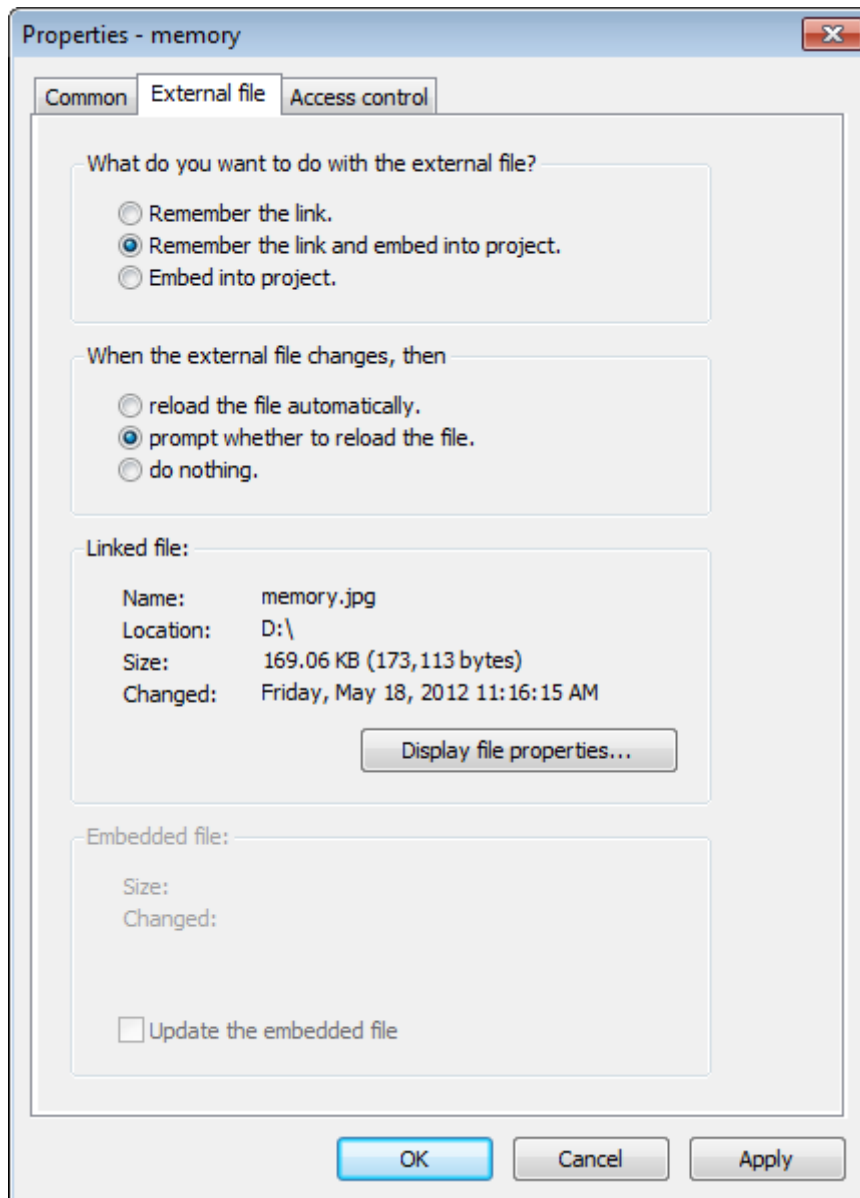


Figure 7-32. Properties Dialog, External File

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.

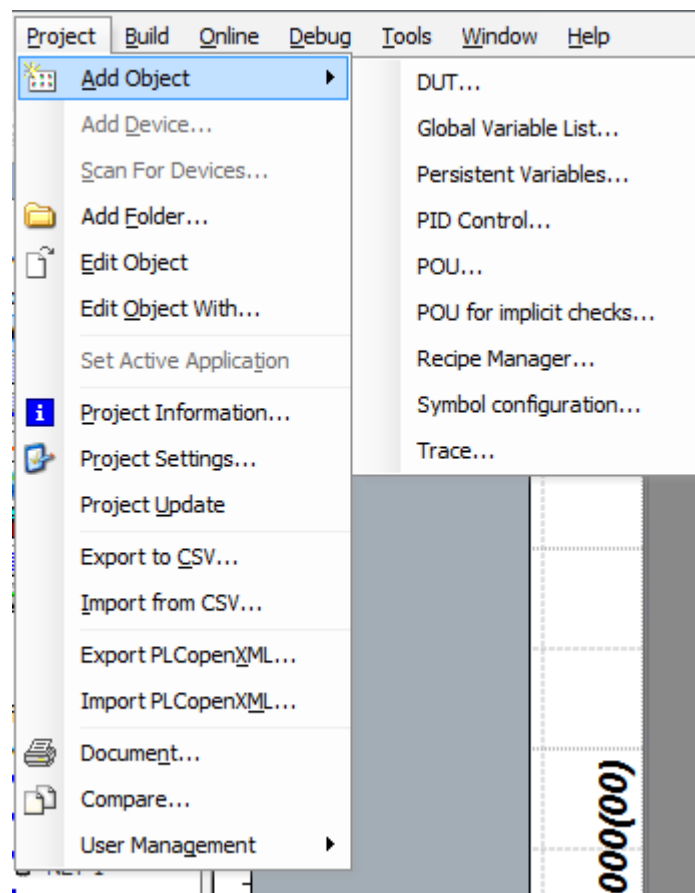


Figure 7-33. “Add Object” Submenu

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 POU's 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 affect 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.

Scan Devices

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

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

Once the search conditions are met, by executing the mentioned command the screen in Figure 7-34 will be displayed, showing the detected devices.

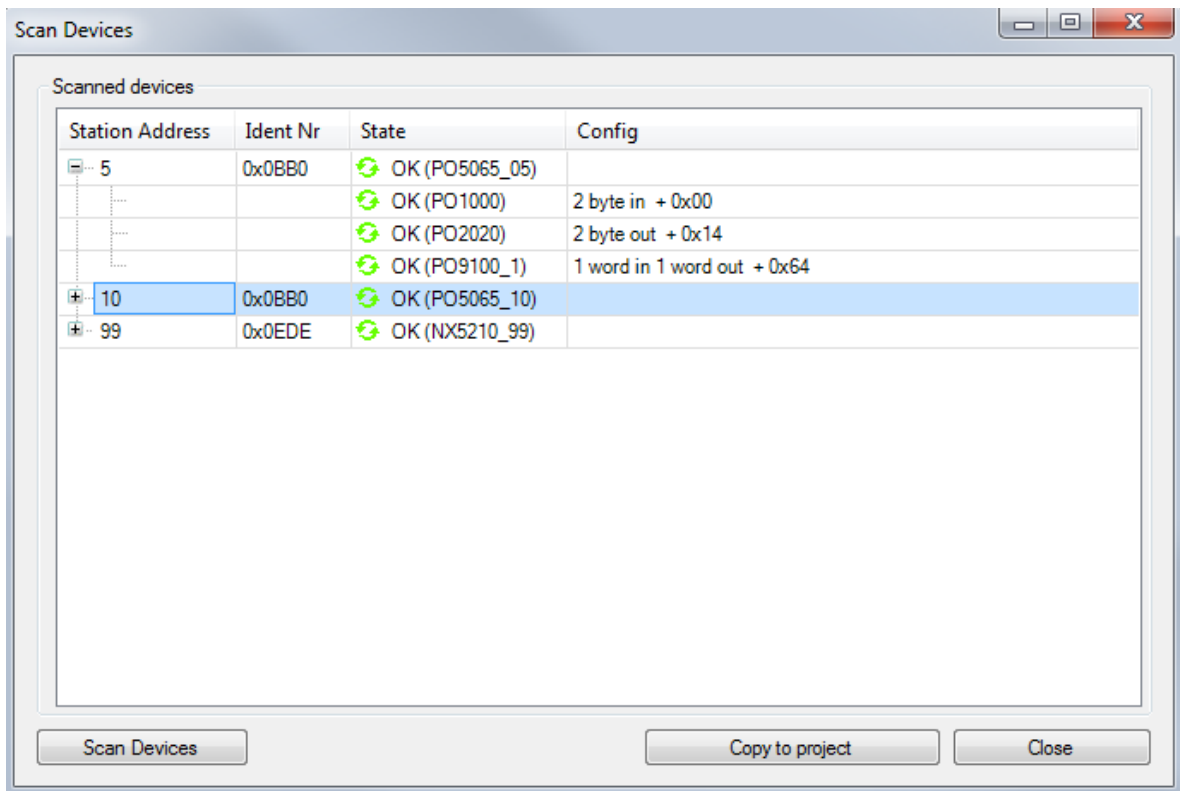


Figure 7-34. Scan Devices Screen

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

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.

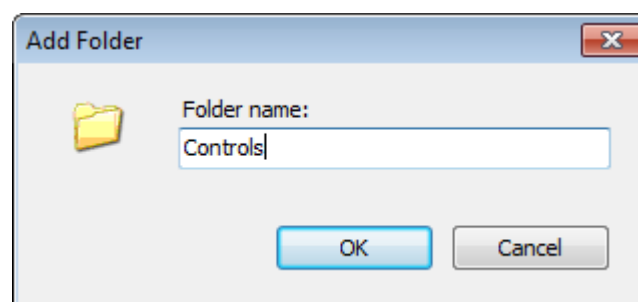


Figure 7-35. Add Folder

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.

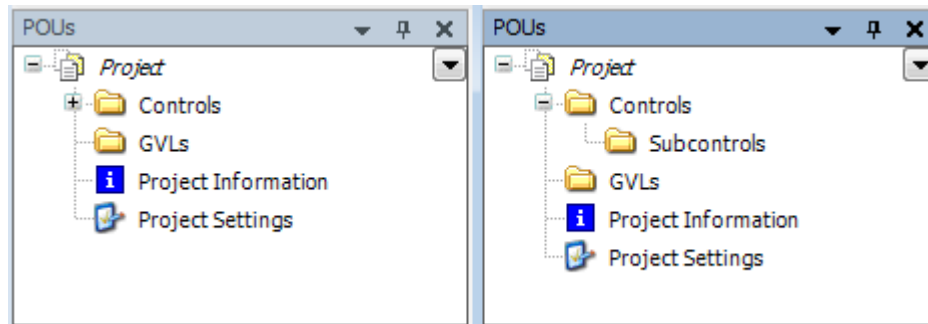


Figure 7-36. Folders in POUs View

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

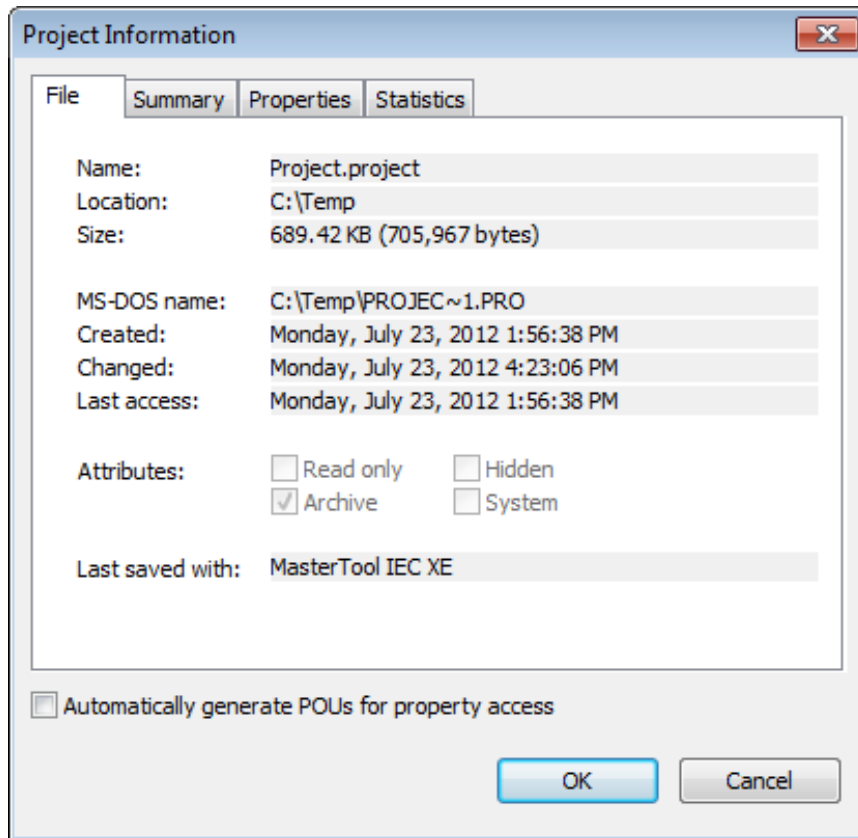


Figure 7-37. Project Information Dialog, File

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

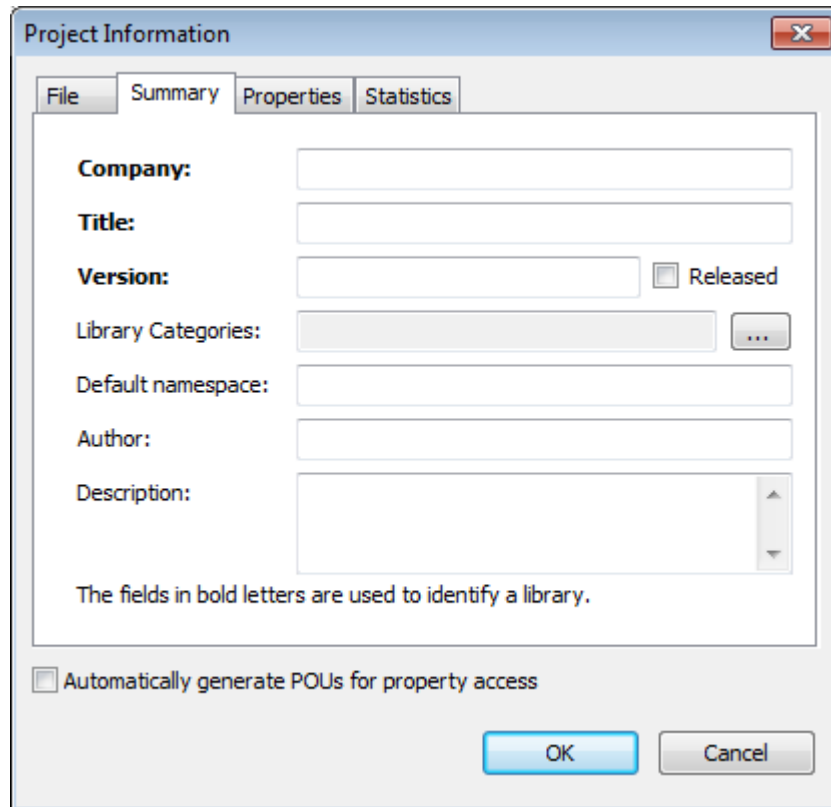



Figure 7-38. Project Information Dialog, Summary

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.

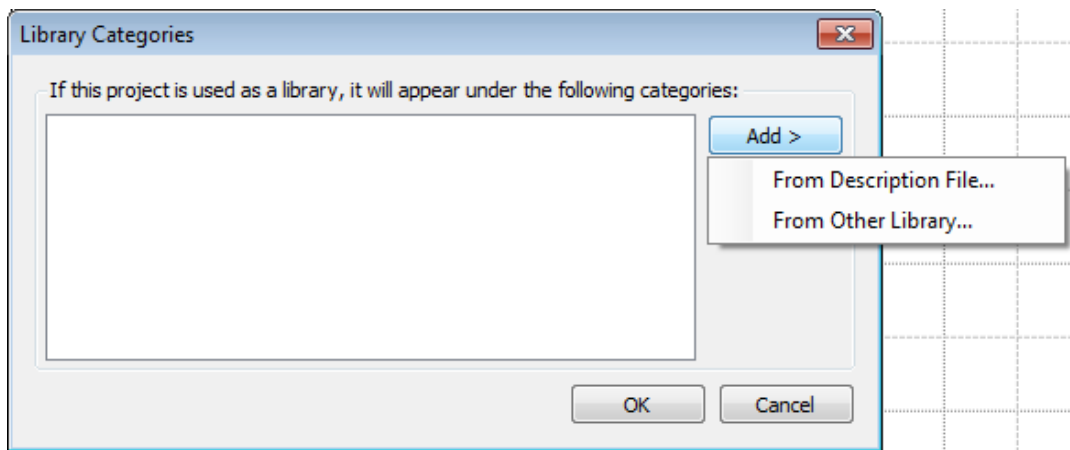


Figure 7-39. Library Categories Dialog

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

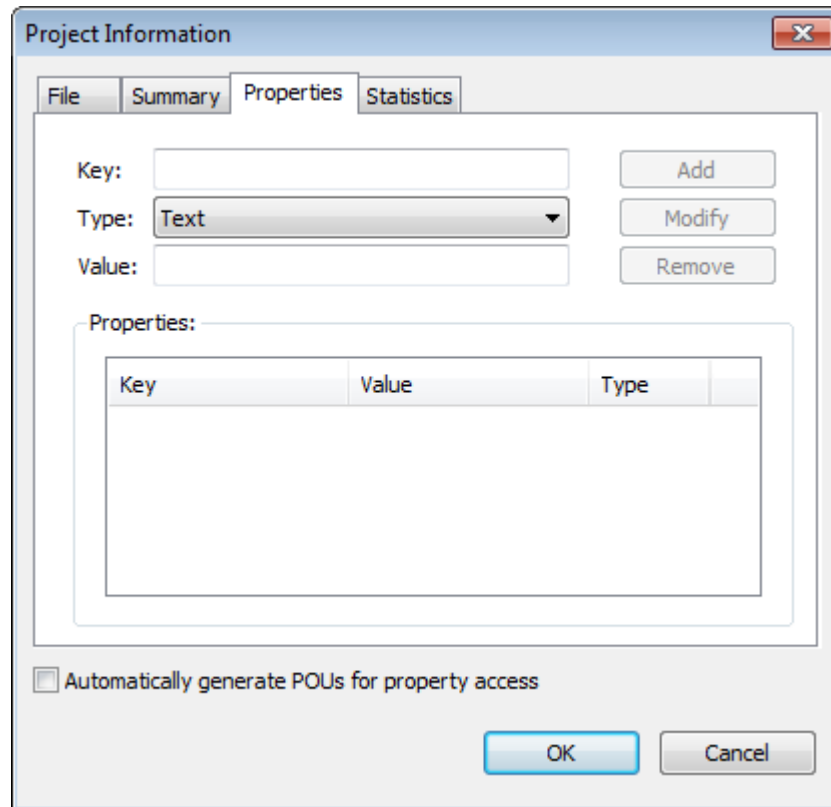


Figure 7-40. Project Information Dialog, Properties

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

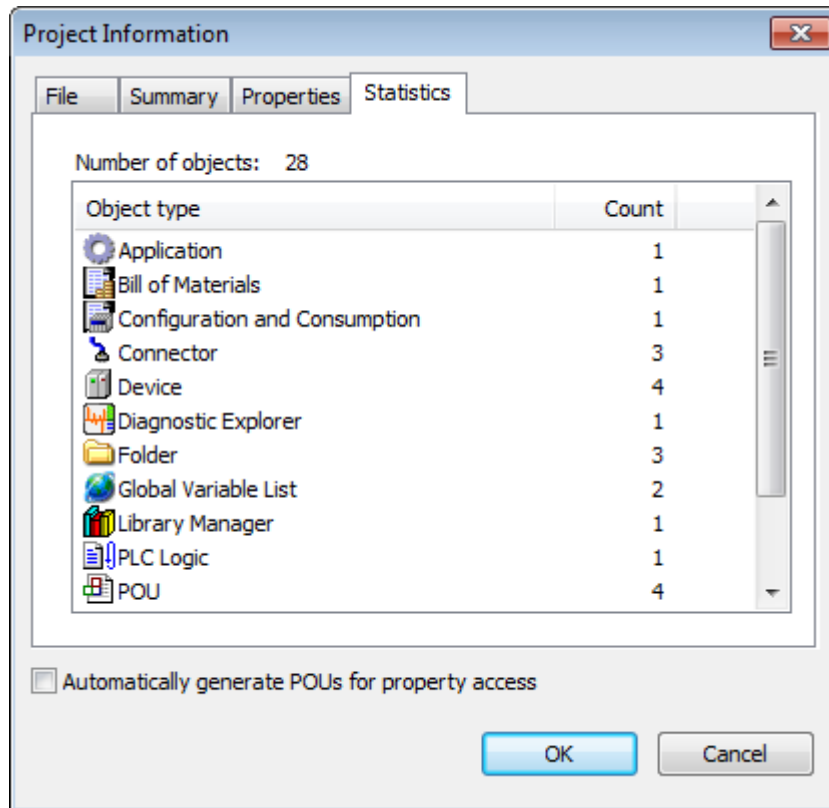


Figure 7-41. Project Information Dialog, Statistics

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.

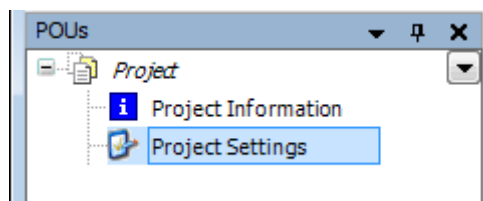


Figure 7-42. Project Settings Object in POUs Window

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

Compiler Warnings

Warning list whose verification in a project compiling is selectable.

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

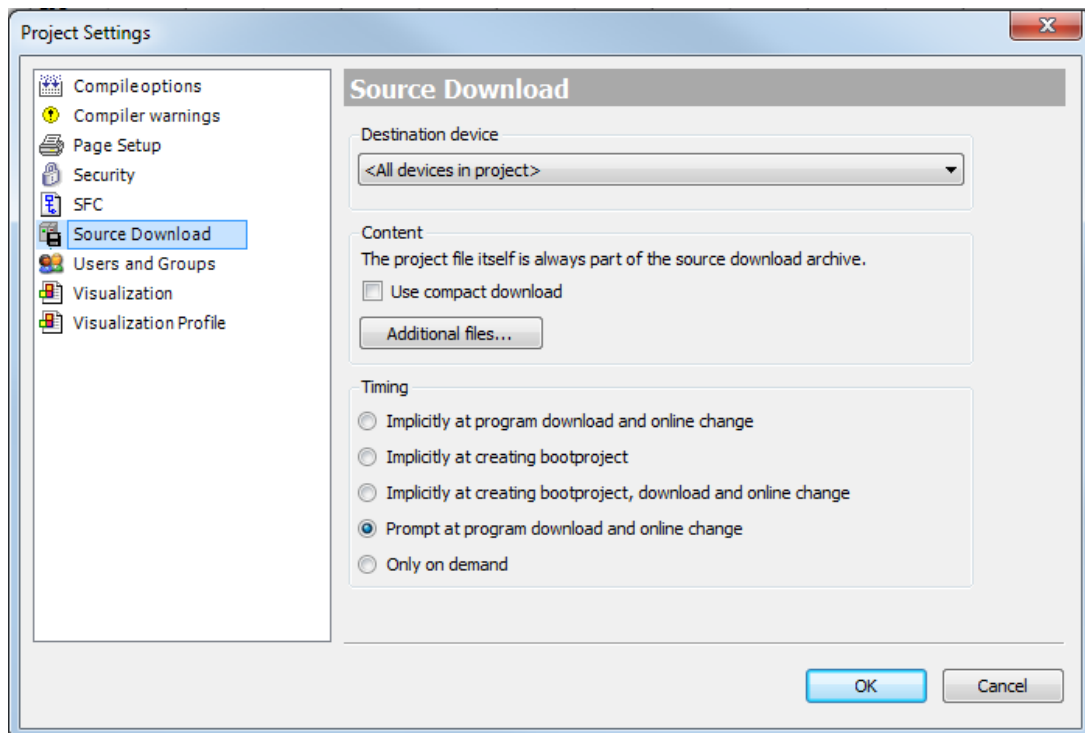


Figure 7-43. Project Settings Dialog, Category Source Download

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 and online change:* The source download is executed at each program download or online change without any user interaction.
 - *Implicitly at creating bootproject:* The source download is executed at each boot project creation without any user interaction.
 - *Implicitly at creating bootproject, download and online change:* The source download is executed at each program download, download or online change without any user interaction.
 - *Prompt at program download and online change:* At each program download or online change the user will be asked whether also a source download should be done.
 - *Only on demand:* The source download has to be initiated by command *Source Download*.

Compile Options

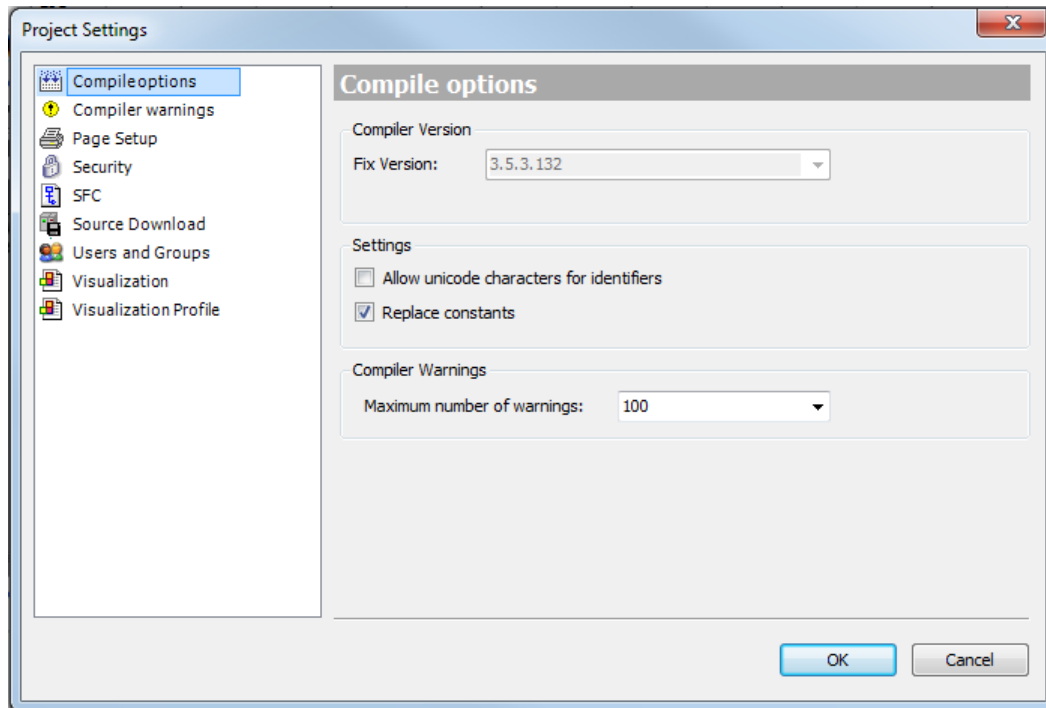



Figure 7-44. Project Settings Dialog, Category Compile Options

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.

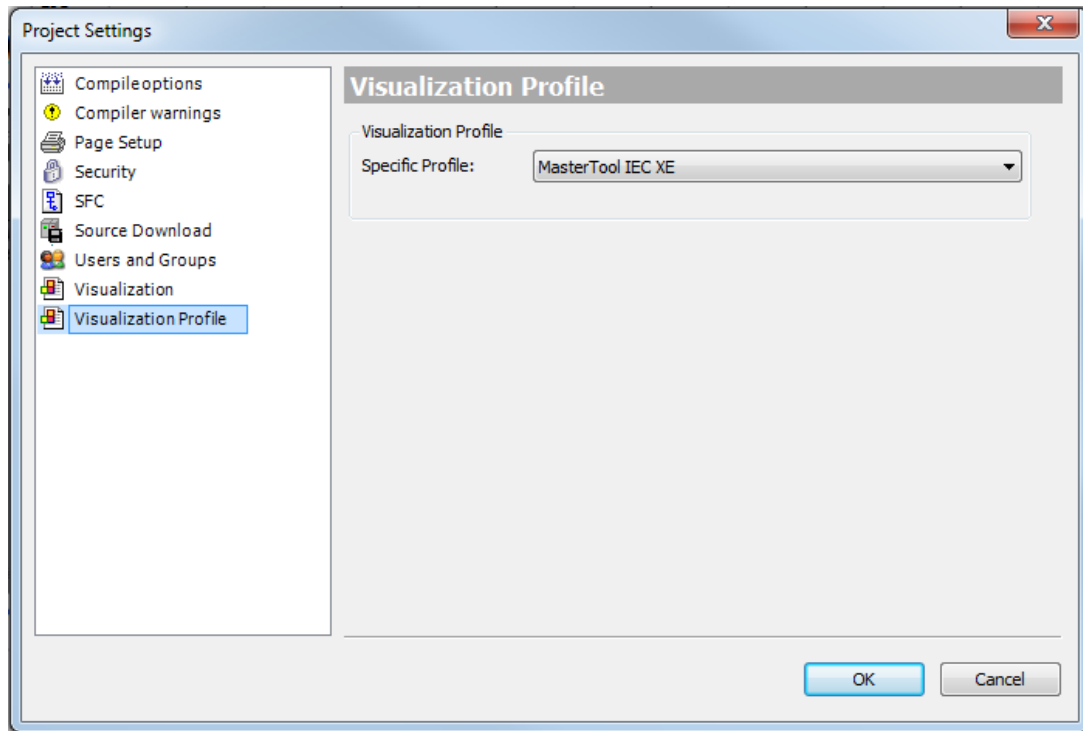
Visualization Profile

Figure 7-45. Project Settings Dialog, Category Visualization Profile

Define here a visualization profiles to be used when the project gets opened. Currently there is the MasterTool IEC XE visualization profile available to Nexto Series CPUs:

Security

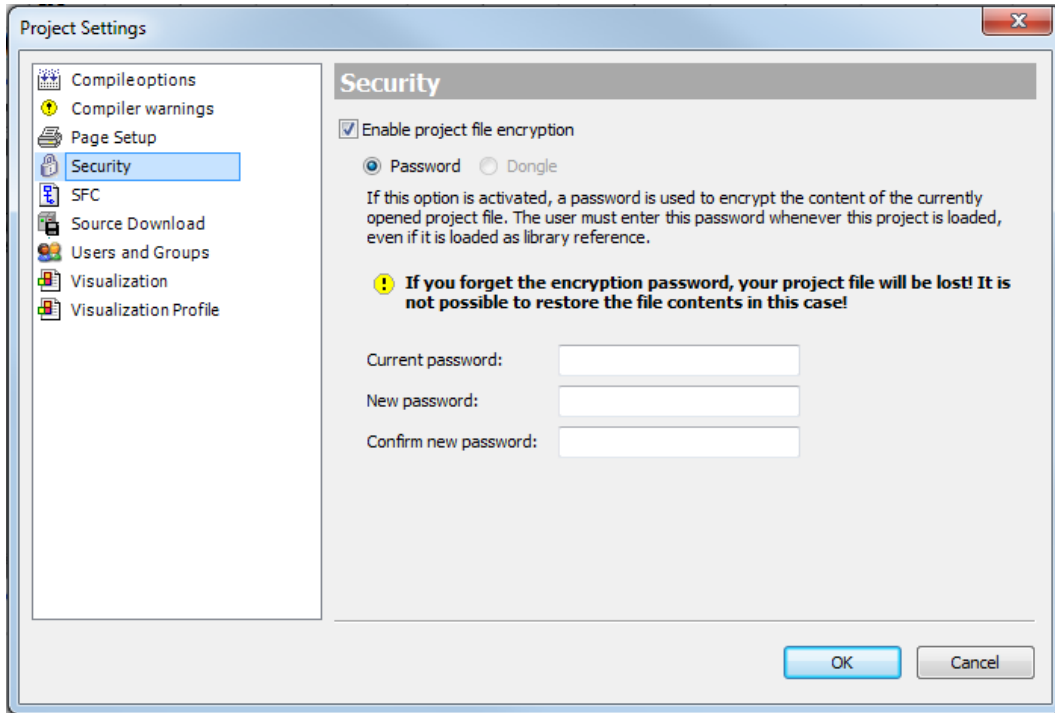


Figure 7-46. Project Settings Dialog, Category Security, Password

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.

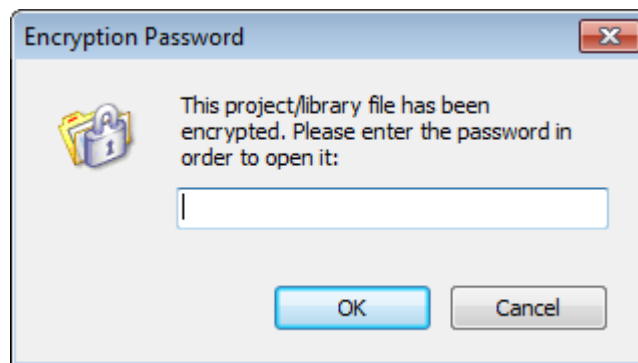


Figure 7-47. Encryption Password

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*.

SFC

It displays a list of variables reserved for use in SFC.

Users and Groups

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

- *Users*
- *Groups*
- *Settings*

For further information, consult **User and Access Right Management**.

Visualization

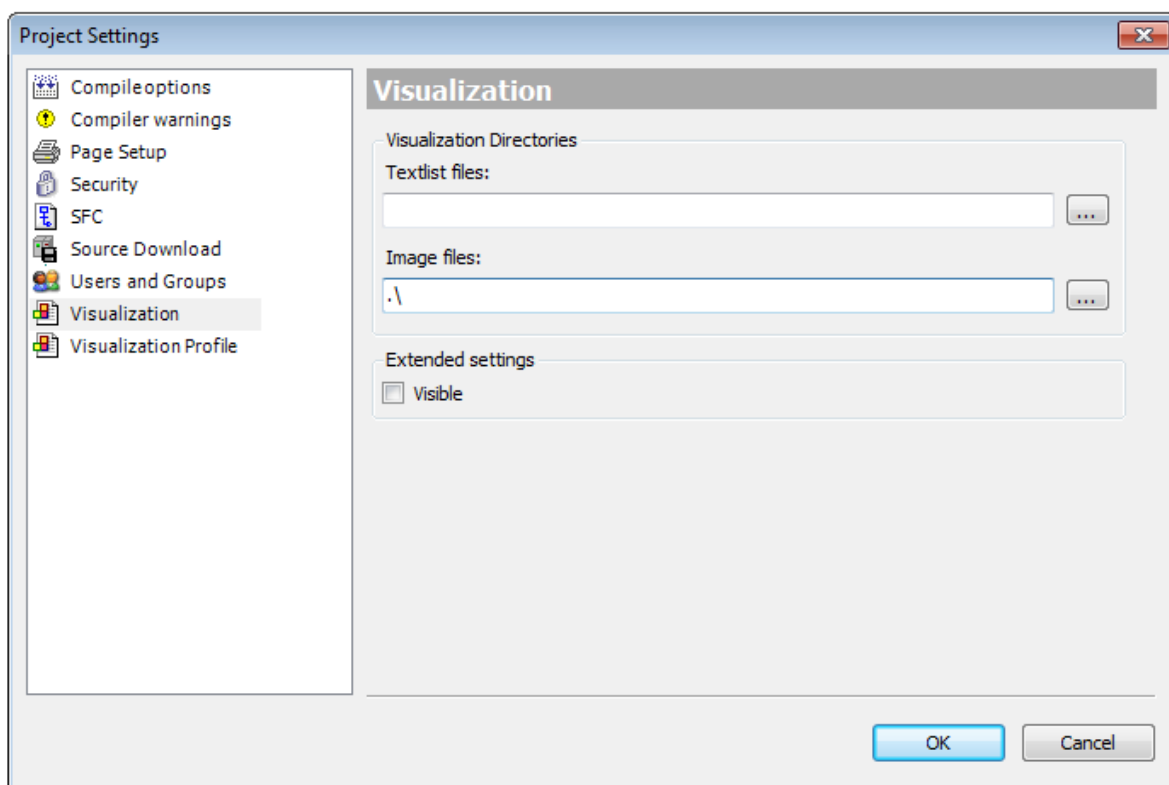


Figura 7-48. Project Settings dialog, category Visualization

Visualization Directories: The default settings for the following directories are defined by the **Options...**

- *Textlist files:* Use button to open the standard dialog for specifying an existing or new directory for language files, which should be available for text and language configuration within the visualization. The files found in this directory will be available for selection when a textlist should be imported, e.g. after having been edited by an external translator. Only one directory can be specified here.
- *Image files:* Use button open the standard dialog for specifying one or several existing or new directory for image files which should be available for the use in the visualizations. If entering several directories, separate by semicolons.

Extended Settings: Can be released by selecting the *Visible* option.

- *Activate property handling in all element properties:* You can set up a display element with a property rather than with a variable IEC those properties in which you select a variable. MasterTool then creates an additional code for the property handling when compiling viewing.

Requirements: Your IEC code contains at least one object of type Property Interface, or a property



Figura 7-49. POU with Property

Project Update

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

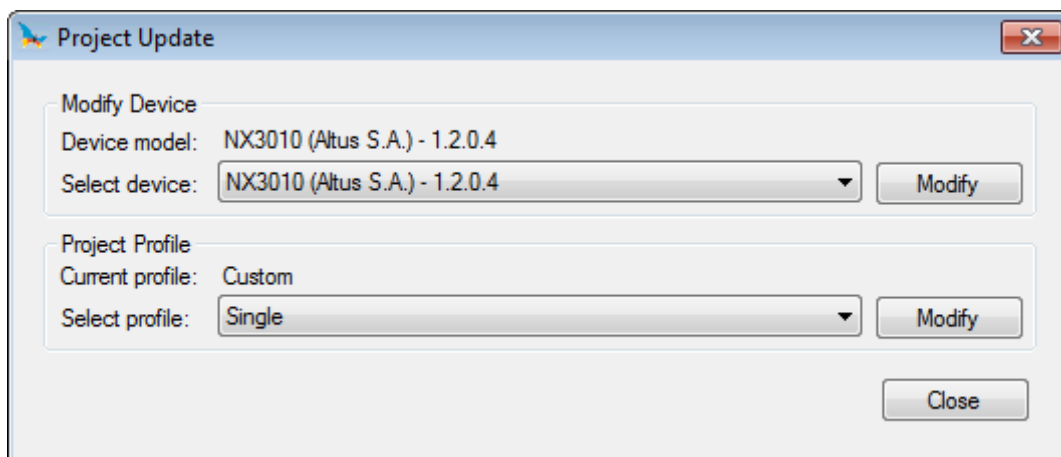


Figure 7-50. Project Update

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.

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 CPU 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 7-51, so all parameters will be kept as in the original project. 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”, so all parameter are going to be reset to the default of the configurator.

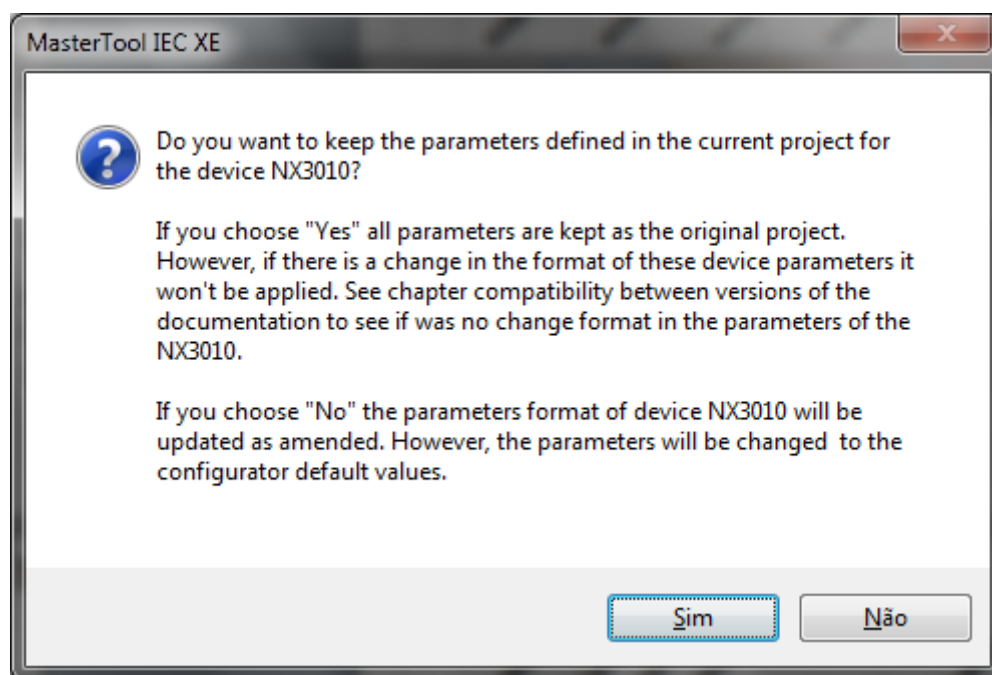



Figure 7-51. Device Update Message

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

Old Projects with PROFIBUS Slaves from Other Vendors


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

When a MasterTool IEC XE version is uninstalled, it removes all installed files, including the device description files installed with the GSD. This way, by uninstalling MasterTool IEC XE version and

installing a new one, if projects with slaves from other vendors created prior to that are opened, they will be displayed in the project tree with the  icon. In these cases, two actions can be taken in order to correctly use the project:

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

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

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

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 to export device configurations and, in many cases, the whole device. When it comes to devices included through the product library, those that can be dragged through the graphic area of the project, only their configurations can be exported. But in case of a device that is attached to these, as a MODBUS or even a PROFIBUS device, they can be completely exported.

The export occurs through the Export to CSV... option of the Project menu or through the Export to CSV... option of a device context menu.

The *Project* menu option exports all the exportable devices and objects from the project, including: POU, GVLs (in or outside folders), configurations of the devices in the racks, MODBUS and PROFIBUS devices. The context menu option, however, exports only the object or devices selected and their children. The export process is recursive and run until all devices associated to a selected device are exported.

There’s a special case where the export can seem less intuitive, it happens when we export selectively a CPU. As all devices connected to the rack are under the CPU’s management, internally, MasterTool considers them as children. So, when we export the CPU through the context menu, all the devices connected to it (seen in the device tree) are exported, as well as all devices present in the graphic area where the racks are.

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 7-52. 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.

ATTENTION:

The export command doesn't export the declaration of the variables that are being used in the devices, mappings or requisitions. Therefore, the import command won't declare these variables if they don't exist in the project.

Some data aren't liable to export:

- The *Process Data* tab of the I/O modules;
- In the *Bus: I/O Mapping* tab, the value of the "Always update variables" selection box;
- In the *Bus: I/O Mapping* tab, bitwise mapping of some modules;
- The memory card access passwords, available through the *CPU General Parameters* tab.

Only POU's objects with ST (Structure Text) language can be exported and imported.

NOTES:

- The devices that can't be exported have the *Export to CSV...* option from the context menu disabled. Similarly, this devices won't be exported if the option from the *Project* menu is used.
- The file generated by the export is a CSV file, this is a sheet file and it's recommended to use the Microsoft Excel program to make any modification in it. Modifications done in any other tool can be susceptible to errors at the time of an import. By saving the CSV file using the Microsoft Excel program, it must be selected the "CSV (MS-DOS)" type. A confirmation message will then be displayed saying that the file may contain features that are not compatible with CSV type files, and it will question if it's desired to maintain the work sheet under work in the current format. This dialog must be answered "Yes".
- This feature is available from the version 1.40 of MasterTool IEC XE to later.

Import from CSV

This command allows to import one or more devices and objects such as POU's, GVL's (in or outside folders) described in the file generated by the export command. The import process is inclusive, not removing devices previously existent in the project.

To import all devices and objects described in the file, the *Import from CSV...* available in the *Project* menu. The devices included through the products library can't be automatically created, therefore, its configurations can only be imported if they are already present in the project.

The partial import occurs when it's desired to import only part of the information present in a CSV file. To do so, it's necessary to choose a device or object present in the CSV file and select the *Import from CSV...* option. As a result, the import process is going to look for all the information about that device in the import file, as well as all information about its children (only valid for devices), due to the import process being recursive. It's important to notice that the information in the CSV file respects not only the type of a device, but also its name. So, a selective import can only be done if the

name of a device present in the import file is the same that the one selected by the time of the import or if there are children of the selected device in the file.

It's worth remembering that the selective import, when executed from the CPU, encompass not only the devices that are visually connected to it (as MODBUS or PROFIBUS devices), but also configurations from devices that are included through the product library. This occurs because all devices connectable to the rack, when included in the project, end up becoming children of the CPU. This behavior is similar to the one informed in the export process described earlier.

By choosing the *Import from CSV...*, the Open dialog box is going to be shown. In it, it's possible to choose the file generated by the export command. After choosing the file and clicking on *Open*, the import will start and all devices described in the file are going to be added to the current project.

The import command acts on current projects. Therefore, there are some situations that need to be handled and their behavior are described below:

- If a device described in the import file doesn't exist in the project, it will be created and it's going to have its configuration as described in the CSV file, unless it's a module.
- If a device described in the import file doesn't exist in the project and it's a module, an error message is going to be displayed, saying that it needs to be created in the project. There's the possibility to remove this module's information from the import file, if the user so wishes.
- If a device already exist in the project, its configuration going to be replaced by the one described in the import file, but only if this device has the same type and is in the same position as the device described in the file. Only the information present in the CSV file is going to be updated, while the rest maintain its values.
- If a device already exists in the project and have the same name, but it isn't in the same position or doesn't have the same type as the device being imported, the import process will be cancelled. It will be informed to the user that the device must be removed from the project or renamed in the CSV file.
- If there's no information to be imported during the import process, the user will be notified.
- If there's an inconsistency in the device data present in the import file, the import will be cancelled and the error will be imported to the user.
- Objects (POU and GVL) that are inside folders will be kept in the folders as in the CSV file. In the import process if these folders do not exist, they will be created and the objects added in it.
- Folders that exist in the project and in the CSV file, but with different "parent folder", will be created with the same CSV file hierarchy, keeping project folders unchanged.

The *Import from CSV...* option from the Project menu allows to import multiple files. The only requirement is that multiple files are selected at the time of the import. The import process will be done once for each file and it's going to have the same behavior as described earlier. It's worth notice that, if an import error occurs, the modifications in the file being processed when the error occurs are going to be undone, but the modifications from the files that were already processed will be maintained.

ATTENTION:

- If errors occur during the import process, appropriate messages will inform the user the reasons for the error occurrence. The error messages vary and it reflects the incompatibility of the imported values with the expected range of values. Once the corrections are made, the import operation must be started by the user again.
- Every time an item from the Bus I/O Mapping tab from a device is updated by the import process, the "Always update variables" selection box is going to be marked.
- To make it possible to import the configuration from a specific device without importing the configuration of its children, it's necessary to remove from the CSV file all the information of these device's children.
- Only POU objects in ST (Structured Text) language can be exported and imported.

NOTES:

- The import command does not create new projects. So, the *Import from CSV...* option from the *Project* menu will be available only if a project is opened.
- The file generated by the export is a file with “.csv” extension, so it’s a sheet file and it’s recommended to use the Microsoft Excel program to do any modification in it. Modifications done in any other tool can be subject to error in the moment of an import.
- This feature is available from MasterTool IEC XE version 1.40 onwards.

Special cases Related to the Import and Export of CSV**Special Columns Exported in Some Devices**

The import process, in some cases, need some extra information about the devices to be imported. Such information is necessary and, somehow, it has been provided by the user during the creation and customization of the project.

When a device is created, it is placed where the user selected. If this device is exported, this information is also exported so that the import process can create it if it doesn’t exist in the project. This information is stored in the *Parent* and *Connector* columns Not all data tables exported in a CSV have this information, but those who need it, have it. These data need not to be changed by the user, unless one wishes to change the behavior of the import targeting some particular purpose.

Special columns using I/O mappings

In the exported file with module mapping(s), there is a column that does not have direct relation with the columns showed on the mapping window. This column is called *Index Offset* and it is necessary to ensure the right behavior between the exported and imported data. This column must be kept without changes after the export process.

Beyond this column, there are two columns named *Channel* and *Type* that must also not be changed, because their content is static in the application. They are present in the exported file to increase the usability when users are managing the file.

Exporting and importing NX1001 and NX1005 modules

These modules have some configurations to enable and disable which kind of mappings will be used and it changes the behavior when exporting and importing data for these modules. These configurations are done through the *Process Data* window when the user need some feature available there.

Select the Outputs		Select the Inputs	
Name	Type	Name	Type
<input type="checkbox"/> HSC Input 00 Command		<input checked="" type="checkbox"/> Digital Inputs	
High Speed Counter Input 00 Command	Struct00	Digital Inputs - Byte 0	BYTE
<input checked="" type="checkbox"/> HSC Input 00 Preset Value		Digital Inputs - Byte 1	BYTE
High Speed Counter Input 00 Preset Value	DINT	<input type="checkbox"/> HSC Input 00 Status	
<input checked="" type="checkbox"/> HSC Input 01 Command		High Speed Counter Input 00 Status	Struct03
High Speed Counter Input 01 Command	Struct00	<input type="checkbox"/> HSC Input 00 Current Value	
<input type="checkbox"/> HSC Input 01 Preset Value		High Speed Counter Input 00 Current	DINT
High Speed Counter Input 01 Preset Value	DINT	<input checked="" type="checkbox"/> HSC Input 01 Status	
<input type="checkbox"/> Pulse-Catch Reset		High Speed Counter Input 01 Status	Struct03
Pulse-Catch Reset - Byte 0	Struct01	<input type="checkbox"/> HSC Input 01 Current Value	
		High Speed Counter Input 01 Current	DINT
		<input type="checkbox"/> Input 02 Period	
		Input 02 Period	DWORD

Figure 7-53. Process Data of a NX1001 Module

When an export process is done, it must be noticed the configuration on this window. The same configuration should be applied previously on the project module where is desired to receive the data from the import process. If the configuration is not the same, some error can happen during the import process. If some error happens because there is difference between file and project configuration, they will be shown as channel incompatibility and the import process will be cancelled. To export and import data from these modules, it is suggested to enable all mapping configurations every time this process is done. It means that the Process Data window will have all boxes checked.

Configuration Data in the Bus Event tab of the CPU

The data from the *Bus Event Configuration* tab of the CPU is saved in the modules that can generate bus events. Therefore, the information is exported through these modules.

For the configuration of the *Bus Event Configuration* tab to be restored, it's only necessary to import the configuration of a module that was selected as bus event generator by the time of the export.

If a module configured as event generator has its data imported in a system, the *Bus Event Configuration* tab of the CPU will be updated with the values of this module, regardless of the values that were configured in this screen prior to the import process. On the other hand, if a module that isn't configured as bus event generator is imported, the data from the bus event configuration screen will remain unaltered.

ATTENTION:

Only one module can be selected as bus event generator. If more than one module is configured that way in the import file, the import process will keep the last imported module to have that configuration as the selected module in the Bus Event Configuration tab of the CPU.

Export PLCopenXML

The PLCopen is an independent entity of manufactures or brands that promote the use of international standards in automation, especially in those cases related to programming and the languages defined in IEC 61131-3 norm. It promotes it through certifications and also through other standards that are based in the IEC 61131-3 norm.

The PLCopenXML is a standard that promotes a manufacturer independent interface to store the information of a project created using the concepts of the IEC 61131-3 norm. This standard predicts a XML file format with schemes determined to store each structure, data and configuration type predicted in IEC 61131-3. These files can be used in any tool that supports that norm. This characteristic allows, for example, that part of a program that was implemented in MasterTool IEC XE to be exported from it to be imported and used in MasterTool Xtorm.

The *Export PLCopenXML* command allows to export the objects of a project created with MasterTool IEC XE in a PLCopenXML format. By running this command, the screen in Figure 7-54 will be displayed.

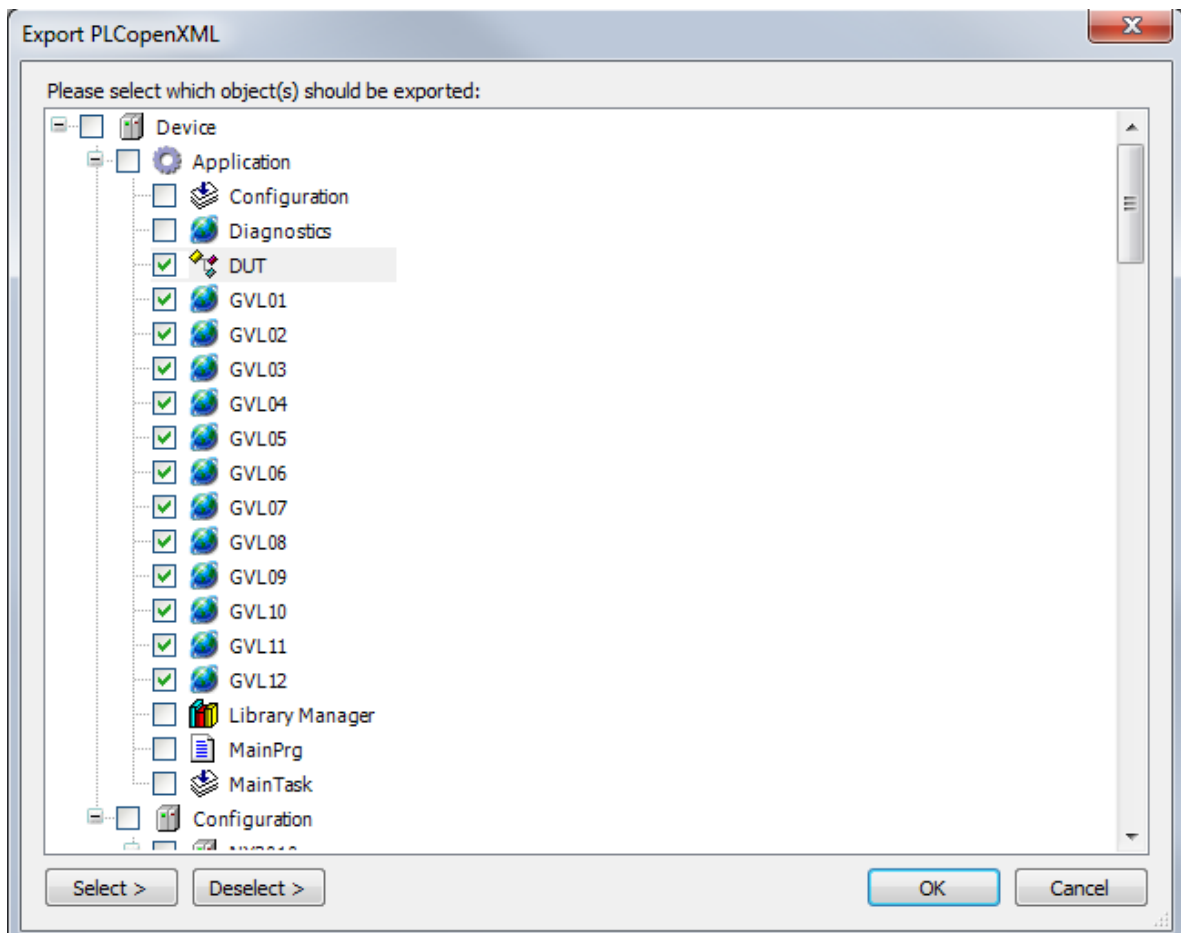


Figure 7-54. Export PLCopenXML Screen

In this screen it's possible to configure individually all objects that are going to be exported. When an object that is parent to others, in the tree structure of the project, is marked or unmarked, all of its children are going to have that same action upon them.

To ease editing it's also possible to use the *Select* and *Deselect* buttons. Through them it's possible to select all objects of a certain type present in the projects, to them mark or unmark them.

After selecting the objects to be exported and press the *OK* button a default save window from the operating system will be displayed. In it, the file name and the path to save it must be chosen. The file is saved with *.xml extension.

Import PLCopenXML

The command *Import PLCopenXML* allows to import into the project the objects from a file in the PLCopenXML format created with the MasterTool IEC XE or other tool that can save files in the PLCopenXML format.

By executing the *Import PLCopenXML* a default “open” window from the operating system will be displayed. The file to be imported must be selected and it must have the *.xml extension. After that, a screen such as the one in Figure 7-55 is going to open.

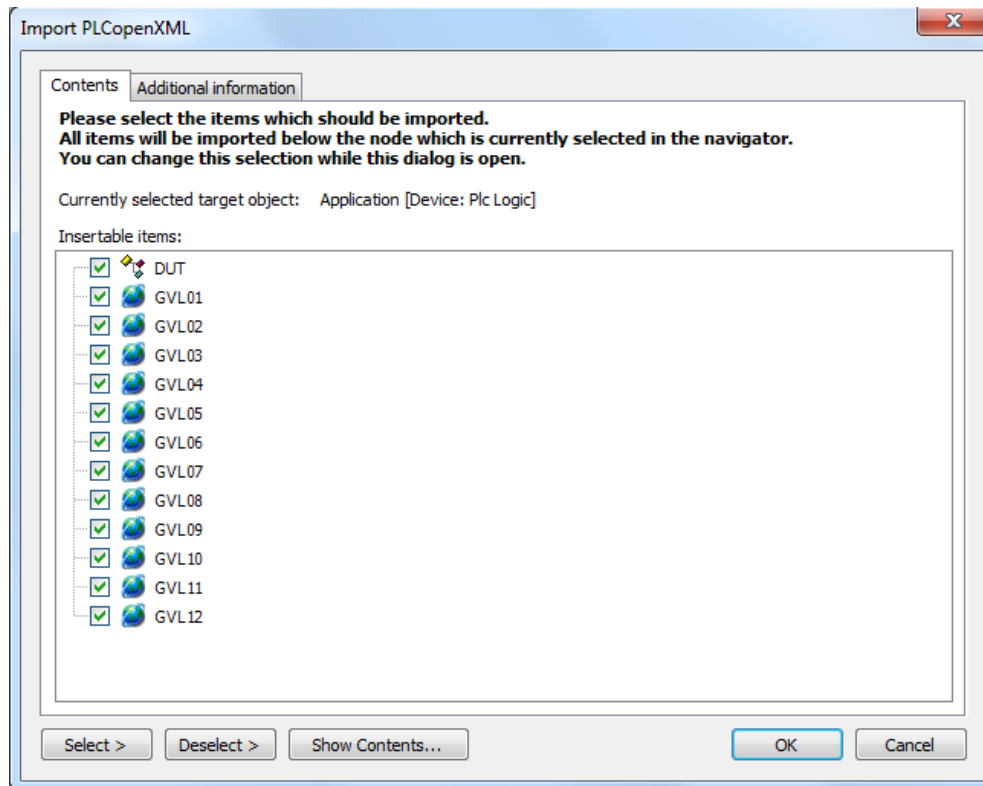


Figure 7-55. Import PLCopen XML Screen

In this screen it’s possible to select which objects available in the file will be added to the project. These objects can be marked individually or with help of the *Select* and *Deselect* buttons that allows to select all the objects of a certain type present in the file to be marked or unmarked. When an object that is parent to others, in the tree structure of the project stored in the file, is marked or unmarked, all of its children are going have that same action upon them.

For the content to be correctly shown it’s necessary that, before executing the *Import PLCopenXML* command, the parent object is selected in the project tree. For example, if POU’s and GVL’s were saved, declared below the *Application* object, this object must be selected before executing the import command. If this is not done, the *Contents* tab will appear empty. Yet the *Show Contents* button can be pressed and will display what is saved in the file.

The *Additional Informations* tab presents information about the saved file, e.g. the tool in which the file was created.

By clicking in the OK button, all selected objects will be added to the project.

Document

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

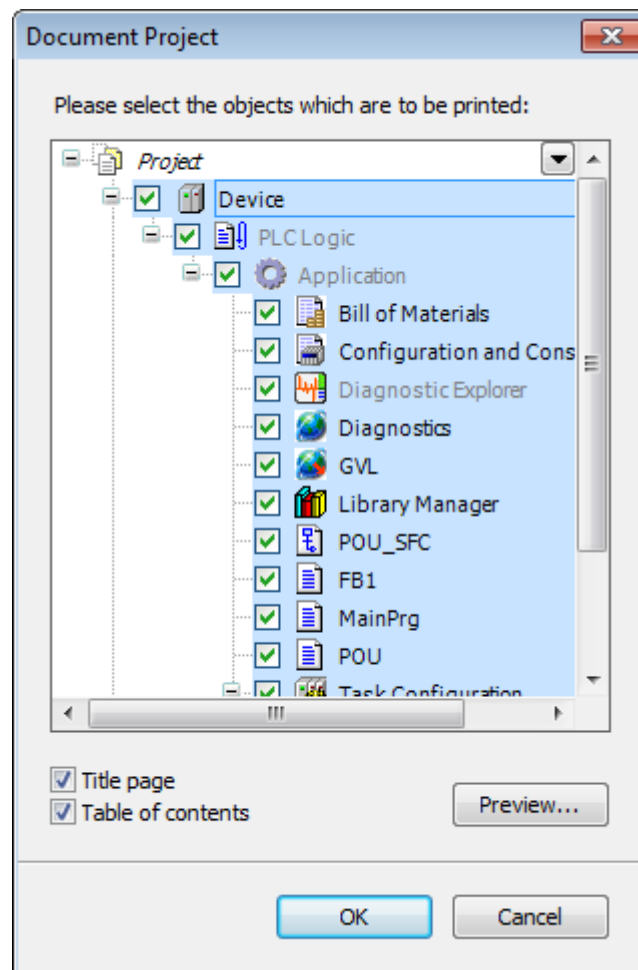


Figure 7-56. Document Project

The project objects in POU's 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.

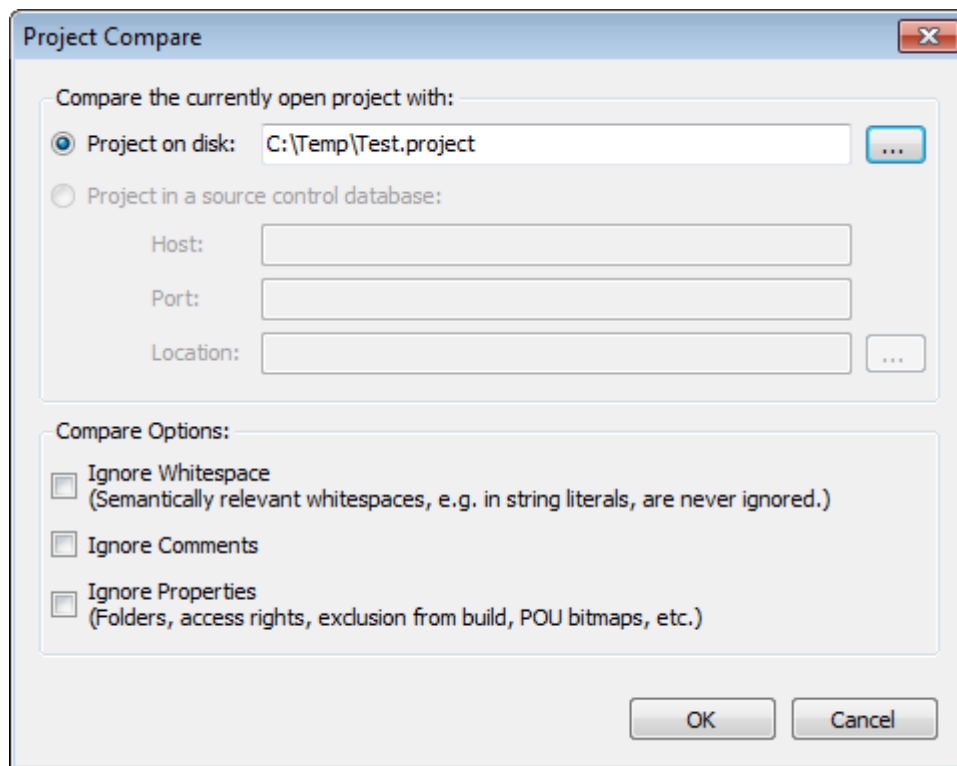


Figure 7-57. Project Compare Dialog

On *Compare the currently open project with...* field, the reference project can be:

- *Project on disk*: by default the path of the actual project itself is entered, so that it will be compared against its version latest saved. You can replace it by modifying the file path after a click on the text field. Alternatively, you may make use of the (...) standard dialog box for browsing after a click on.

On *Compare options* field one or several of the following options concerning the comparison can be activated:

- *Ignore Whitespace*: Discrepancies due to a different number of blanks will not be mentioned.
- *Ignore Comments*: Comments will be excluded from comparison.
- *Ignore Properties*: Object properties will not be compared.

After closing the dialog *Project Compare* with *OK*, the comparison is executed according to the settings.

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*.

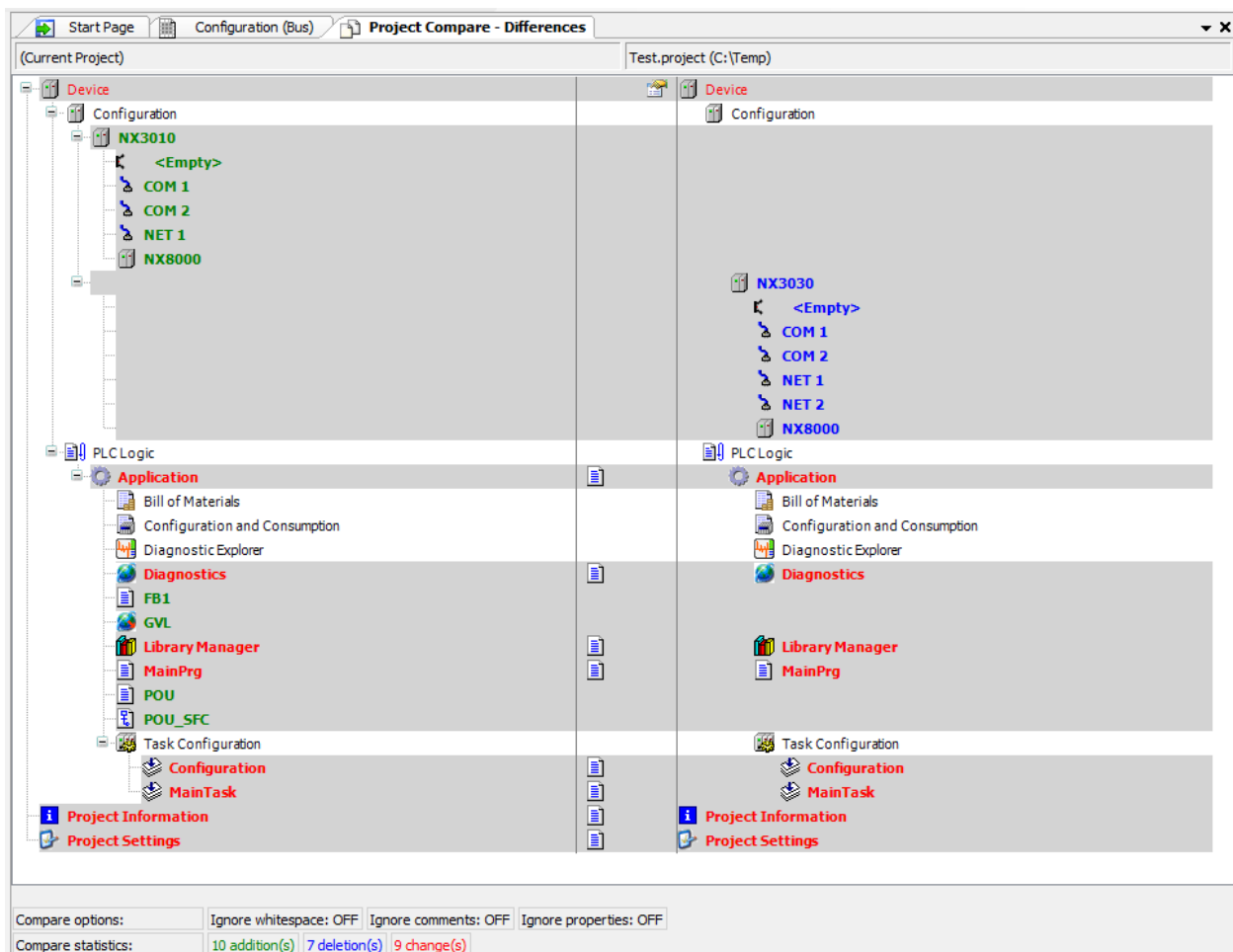





Figure 7-58. Comparison Result

On top of the new window a toolbar is placed at your disposal, followed by a title bar and a sub-window. The title bar as well as the sub-window itself is subdivided into a left part representing the current project and the right one representing the reference project. The corresponding file paths are displayed in the title bar whereas the sub-window shows the two device trees associated to the projects.

Therein, you see the name of identical units displayed in black with no further remarks. Otherwise, the name of an unit is displayed:

- In bold blue, if it exists in the reference project only; instead of its name a gap will be inserted at the corresponding place in the device tree of the actual project.
- In bold green: if it exists in the actual project only; instead of its name a gap will be inserted at the corresponding place in the device tree of the reference project.
- In bold red in both parts of the window, if there are differences in the two versions of the unit concerning the implementation. In addition the unit name is followed by .
- In red, if the differences of the versions are related only to properties  or access rights .

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 7-59). Press in the <BACKSPACE> key to return to the survey.

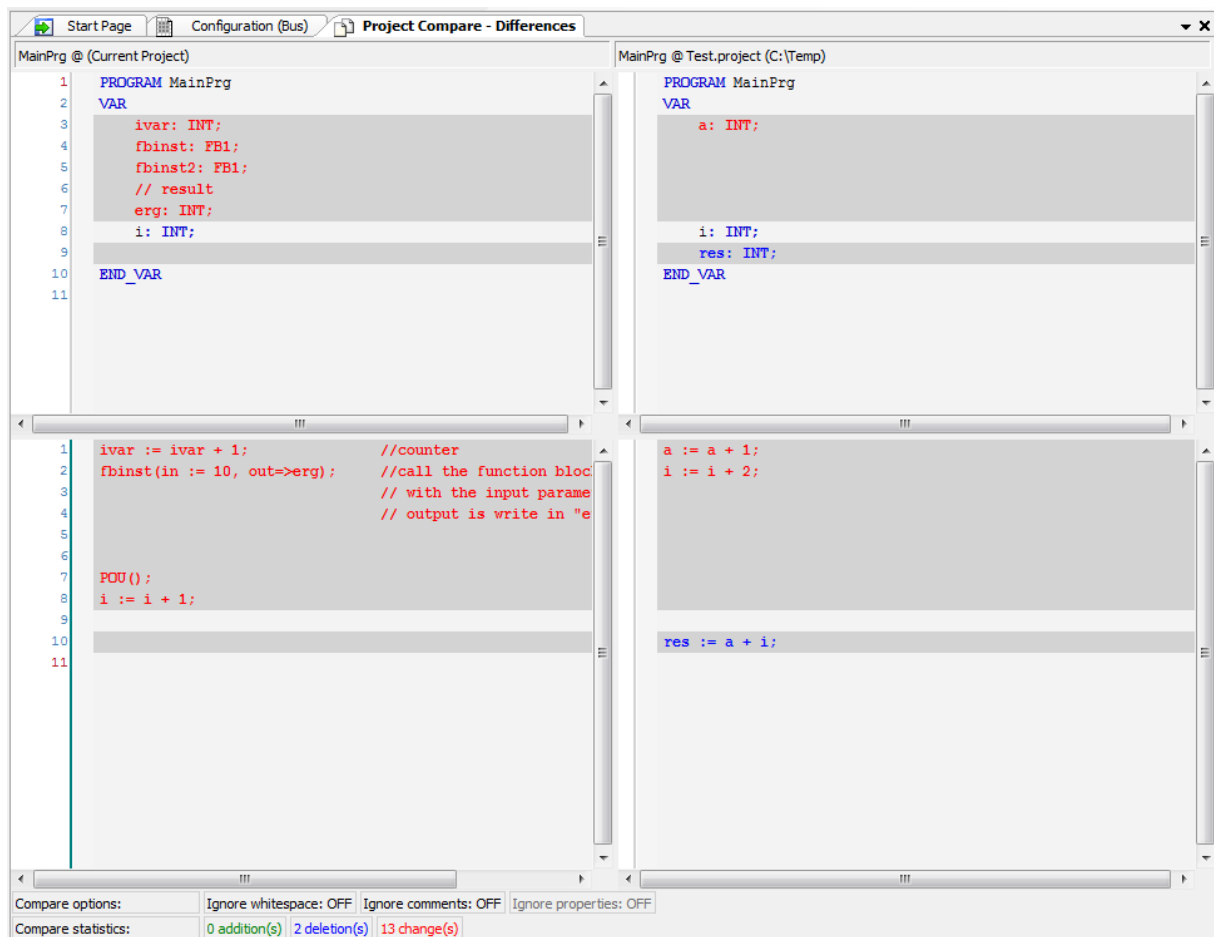


Figure 7-59. Example of Detailed Comparison Result

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**.

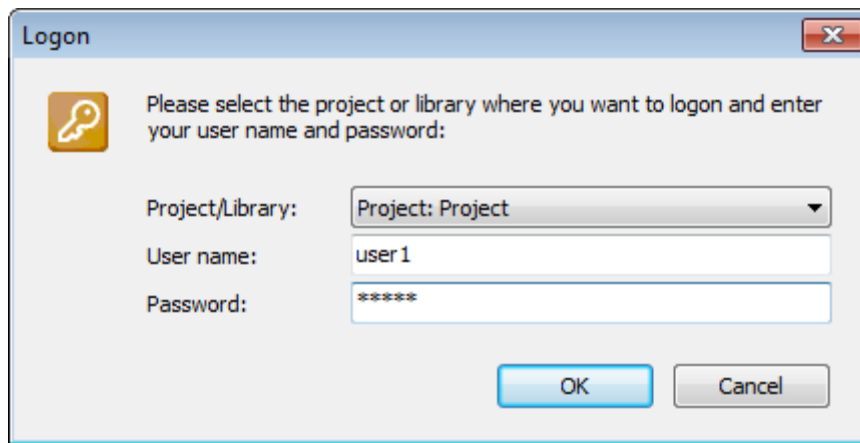


Figure 7-60. Logon

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:

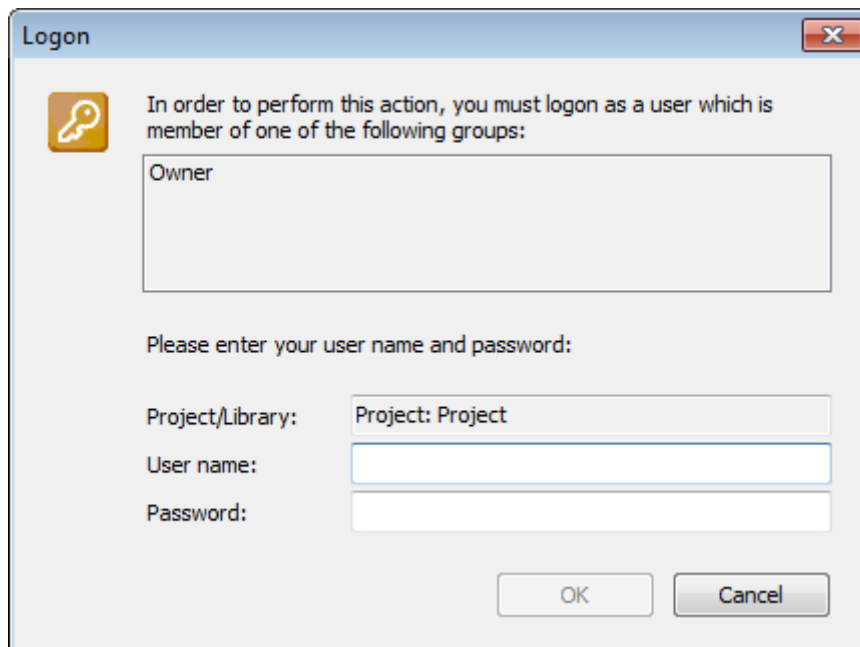


Figure 7-61. Logon Dialog on a Non-Permissible Action

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.

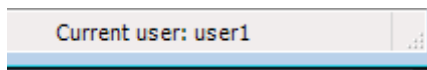


Figure 7-62. Status Line

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.

For further information check **User and Access Right Management**.

Recipe Menu

Visible only when the editor of the *Recipe Definition* object is active, the *Recipe* menu provides commands to work with variables that were added in the *Recipe Definition* object.

The provided commands are:

- Add New Recipe
- Remove Recipe
- Load Recipe
- Save Recipe
- Insert Variable

When the user is logged, a few more commands are added to the menu, they are:

- Read Recipe
- Read and Save Recipe
- Write Recipe
- Load and Write Recipe

NOTE: All commands from the *Recipe* menu are present in the context menu in the editor of the *Recipe Definition* object. The detailed explanation for each of them can be found in the **Recipe Manager Editor** and **Recipe Definition Editor**.

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 affect the network in which the cursor is currently positioned.

A commented-out network will be displayed according to the options set for comments and will not be noticed in program processing.

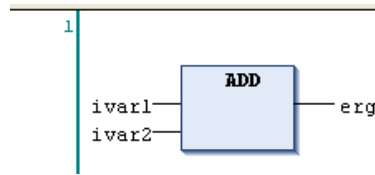


Figure 7-63. Normal State

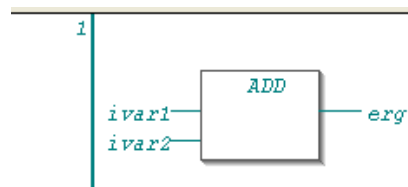


Figure 7-64. Comment State

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: `-[RR]`

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 "???".

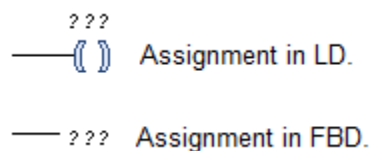


Figure 7-65. Assignment in LD/FBD

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.

1	PROGRAM	ILInsertAssignment
2	VAR	
3		iVar: INT ;
4		bResult: BOOL ;
5	END_VAR	

1	LD	iVar
	ADD	1,
		4
	GT	12
	ST	bResult

Figure 7-66. 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. 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.

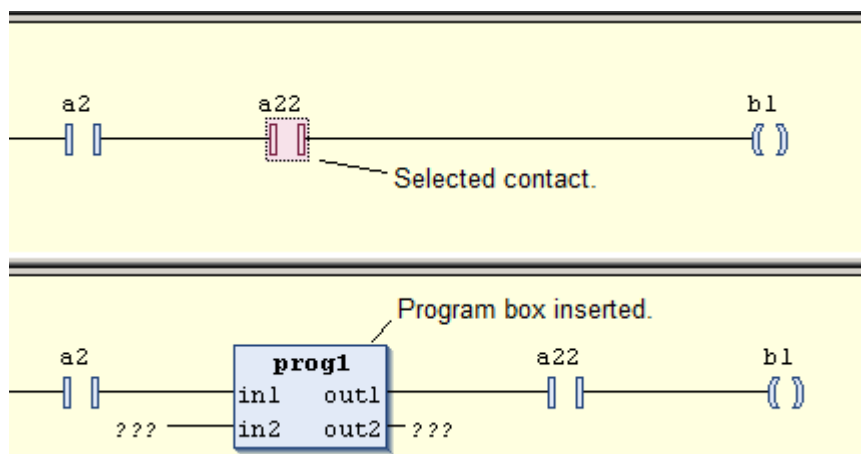


Figure 7-67. Insert Function Block

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 ⇄.

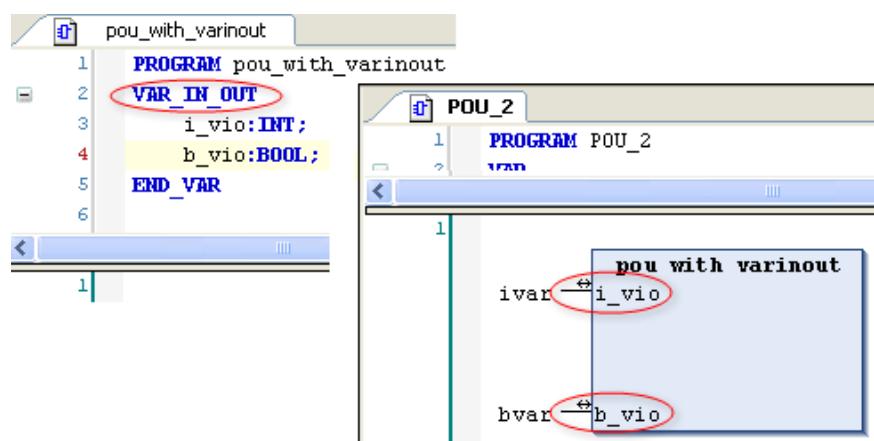


Figure 7-68 Display of VAR_IN_OUT Parameters

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 7-69. 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 Box with EN/ENO

Symbol: 

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

This command is used to insert a box element with the insertion of EN and ENO input and outputs in a network. Besides this difference, this command work in a similar way as the *Insert Box* command.

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.

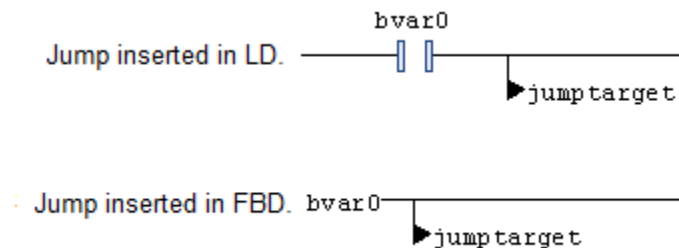


Figure 7-70. Jump Inserted

```

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

1  LD      bVar
   JUMPC  jump target
2
   jump target:
   LD      bVar2

```

Figure 7-71. 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.

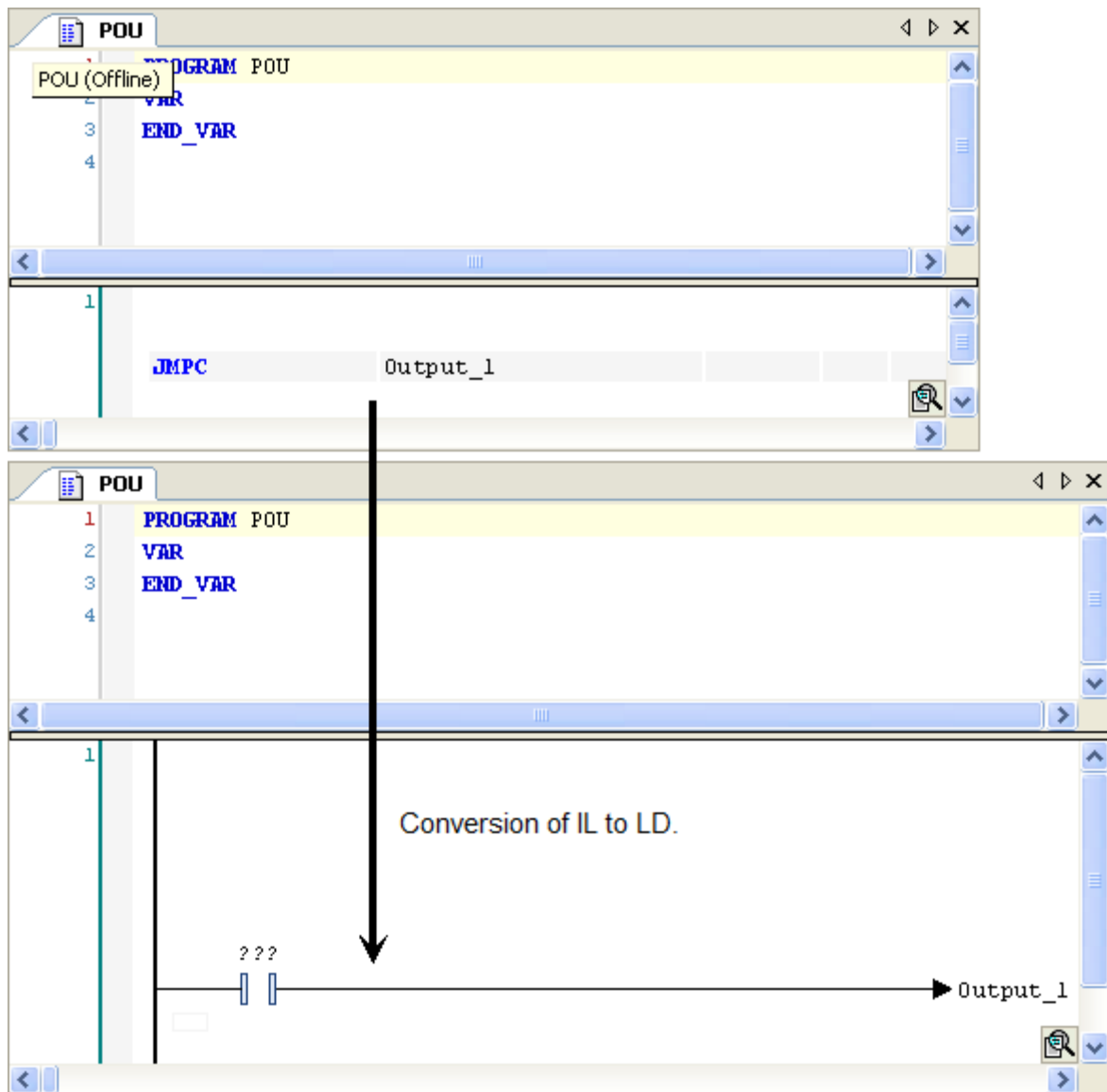


Figure 7-72. JMP Operator Conversion

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:** 

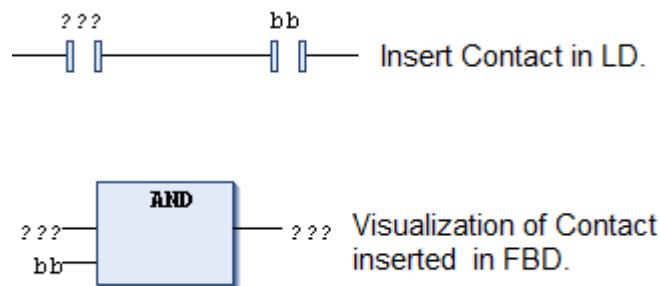
This command is used to insert a set coil. Alternatively, the combination of the commands *Insert Coil* and *Extras Set/Reset* can be used to insert a set coil.

*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 7-73. 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.

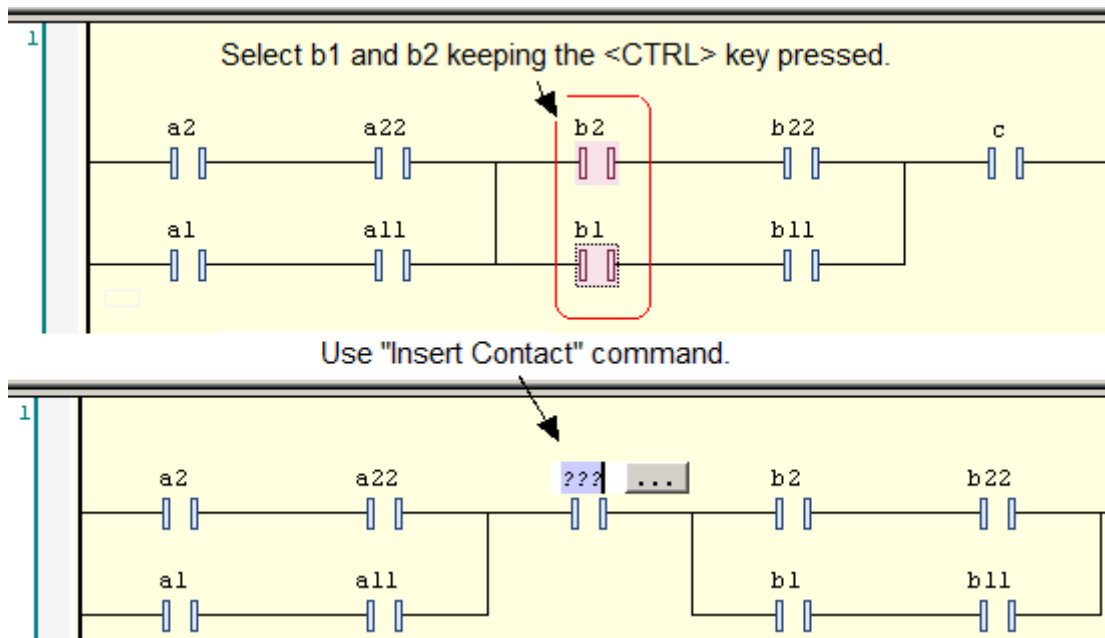


Figure 7-74. Inserting Contacts

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:** 

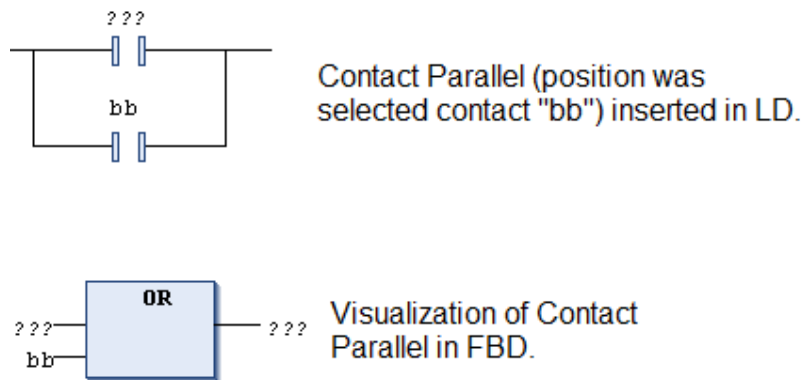
This command is used to insert a negated parallel contact. Alternatively, a combination of the commands *Insert Contact Parallel (Below)* and *Negation* can be used to insert a negated parallel contact.

*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 7-75. 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>+

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:

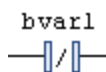


Figure 7-76. Negated Contact

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.

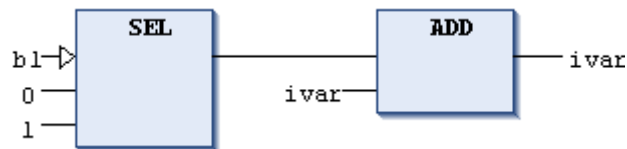


Figure 7-77. Edge Detection at SEL Operator

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.

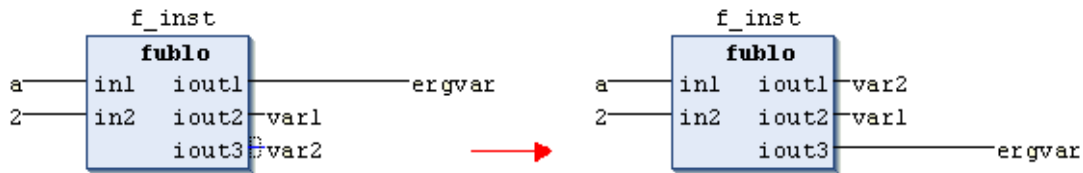


Figure 7-78. Change Output Connection

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.

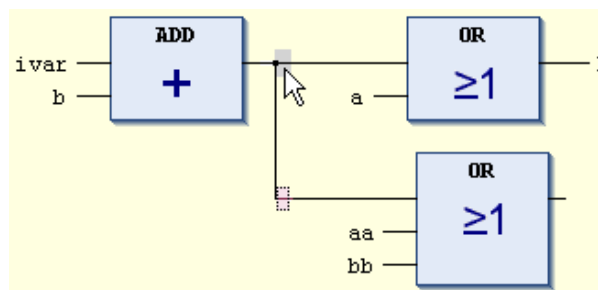


Figure 7-79. Sub-network Markers in FBD Network

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

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**.

Update Parameters

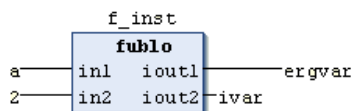
Default Shortcut: <CTRL>+<U>

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.

```
FUNCTION_BLOCK fublo
VAR_INPUT
    in1:INT;
    in2:INT;
END_VAR
VAR_OUTPUT
    iout1:INT;
    iout2:INT;
END_VAR
```



2. Function block modified, added output.

```
FUNCTION_BLOCK fublo
VAR_INPUT
    in1:INT;
    in2:INT;
END_VAR
VAR_OUTPUT
    iout1:INT;
    iout2:INT;
    iout3:INT;
END_VAR
```

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

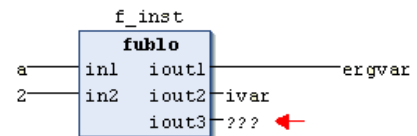


Figure 7-80. Update Parameters in FBs

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.

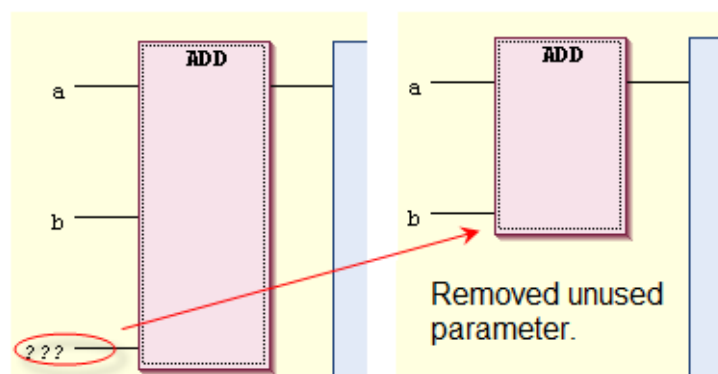


Figure 7-81. Remove Unused Parameters in FBs

*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.

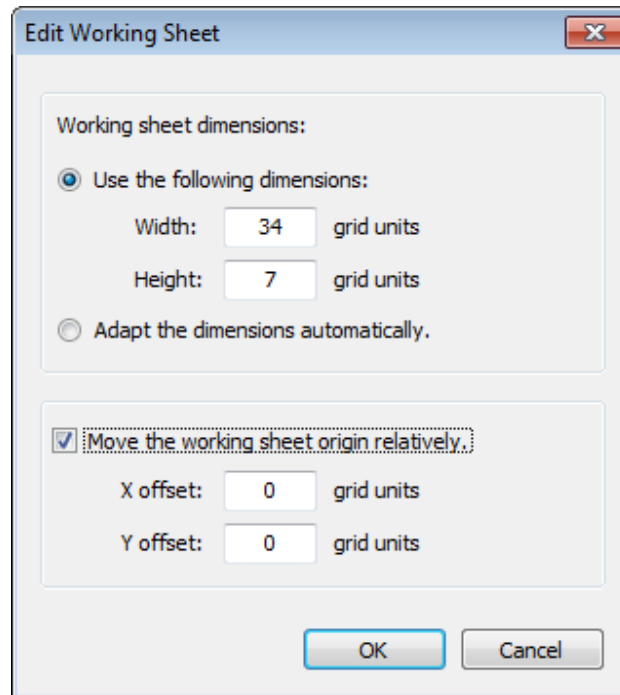


Figure 7-82. Edit Working Sheet

- *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).

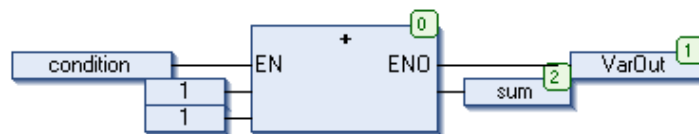


Figure 7-83. ADD-box with EN/ENO

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 7-84 shows how the ENO value can be used for further blocks.

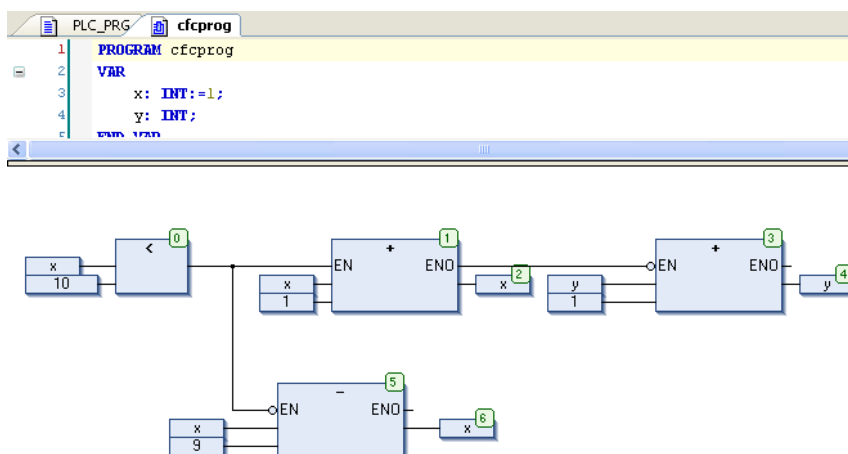


Figure 7-84. Use of EN/ENO

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: ~~x~~

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: ~~R~~

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 7-85. 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: ~~S~~

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 7-86. 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.

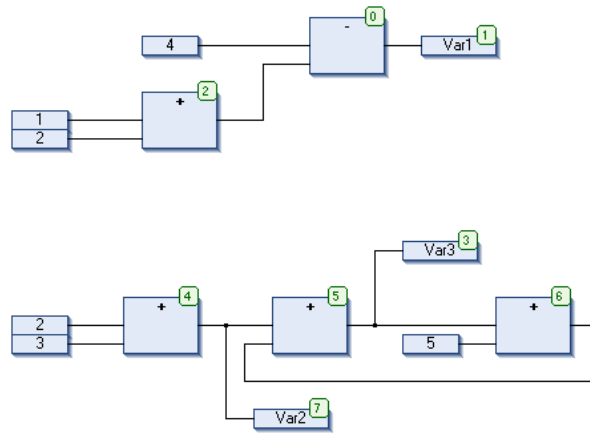


Figure 7-87. Order by Topology

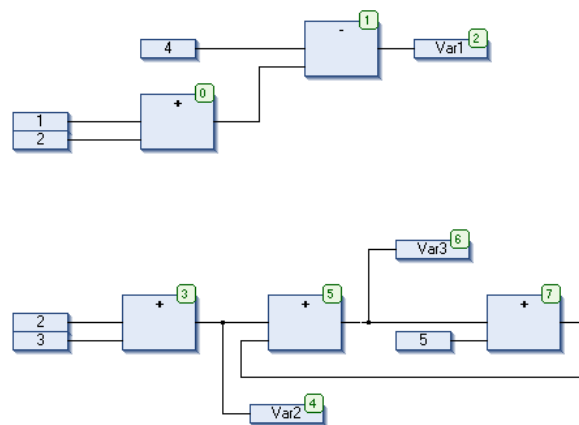


Figure 7-88. Processing According to Data Flow

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:

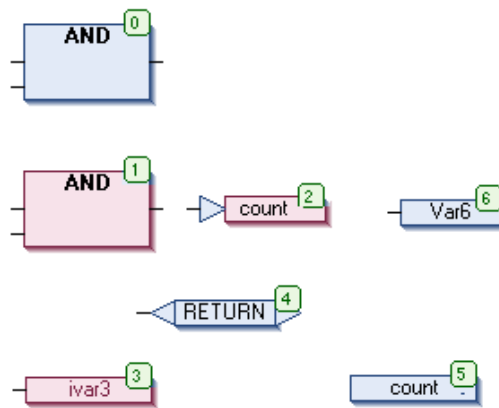


Figure 7-89. Sequence Before

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:

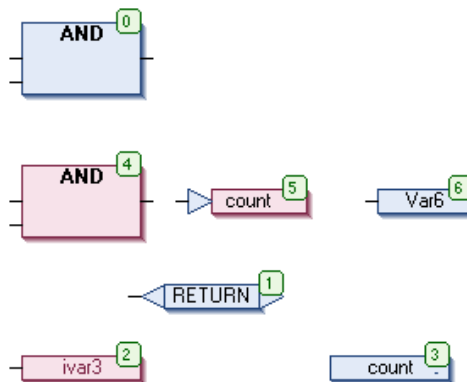


Figure 7-90. Sequence Afterwards

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.

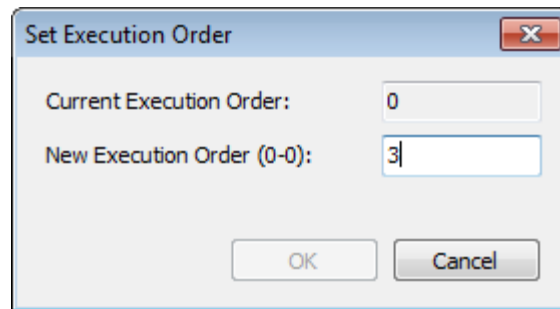


Figure 7-91. Set Execution Order

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.

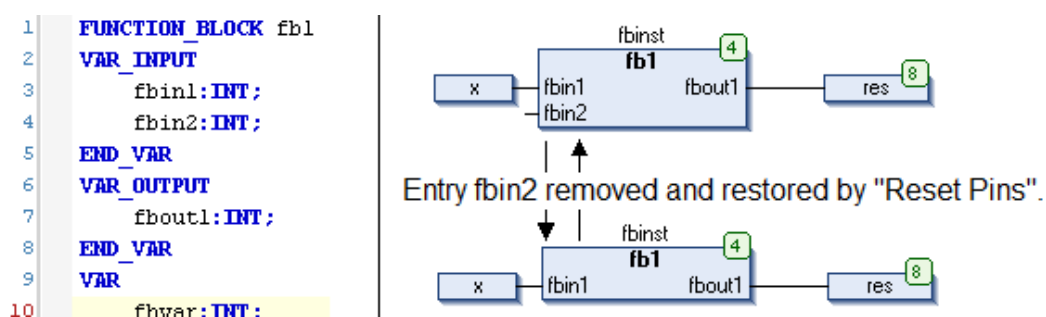


Figure 7-92. Reset Pins

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.

Add Entry Level

Symbol: 

This command is used in the SFC editor to add an Entry Action to an already selected step. This command is effective only when the selected step doesn't have an Entry Action already defined.

The Entry Action runs only one time, when the transition from the previous to the step where the Entry Action were added happens.

By adding an Entry Action, the step where it was added starts showing a square symbol with the E letter in the bottom left side of the step.

Add Exit Action

Symbol: 

This command is used in the SFC editor to add an Exit Action to an already selected step. This command is effective only when the selected step doesn't have an Exit Action already defined.

The Exit Action runs only one time, when the transition from the step where the Exit Action were added to the next one happens.

By adding an Exit Action, the step where it was added starts showing a square symbol with the X letter in the bottom right side of the step.

*Insert Step Transition***Symbol:** 

This command is used in the SFC-Editor to insert a step and a transition before the currently selected position.

The positioning (sequence) of the new step and transition depends on whether a step or transition have been selected when performing the insert command. Automatically the sequence step-transition-step-transition-... will be kept. See Figure 7-93 for examples.

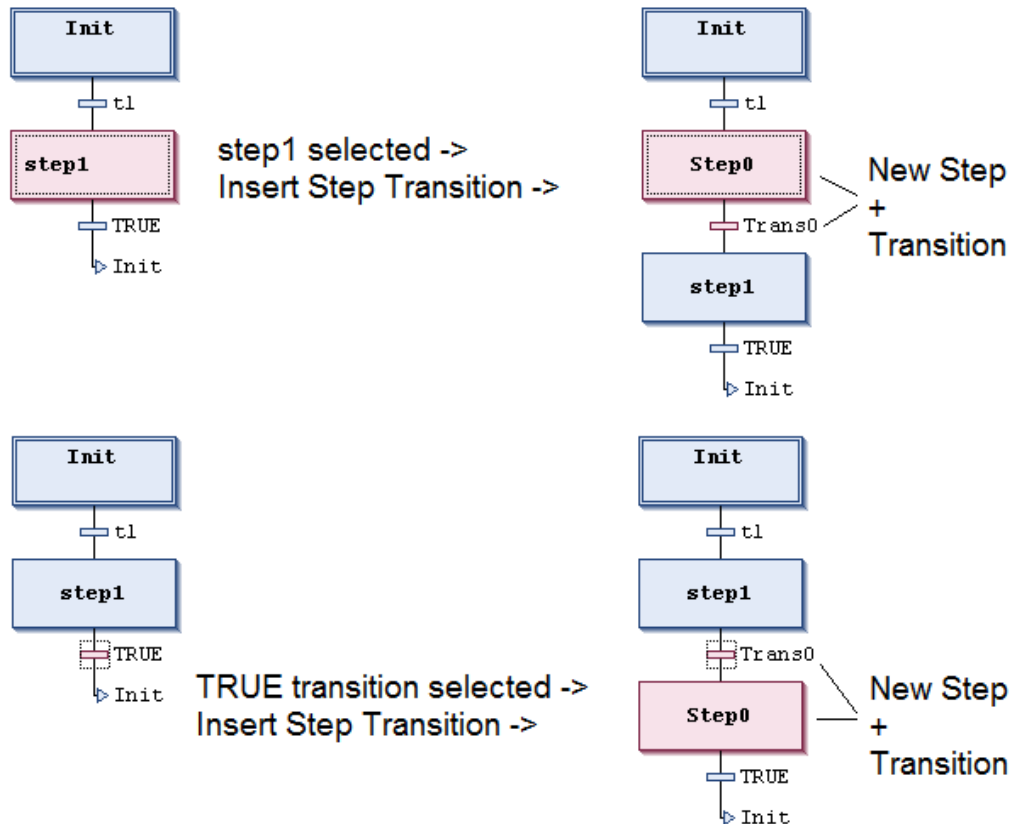


Figure 7-93. Step and Transition Inserted

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-step-transition-... will be kept.

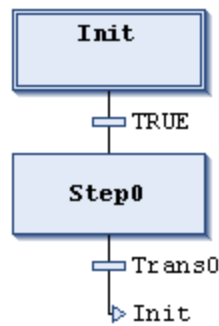


Figure 7-94. Step and Transition Inserted

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.

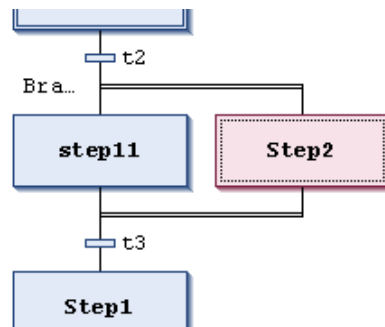


Figure 7-95. Parallel Branch

Example of Alternative Branch

In Figure 7-96 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.

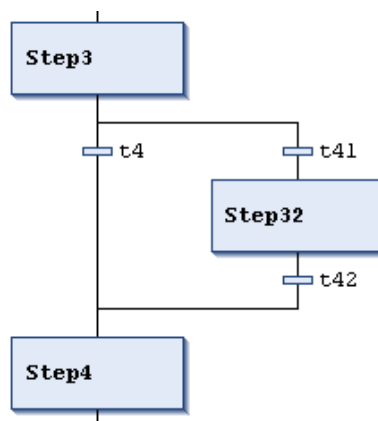


Figure 7-96. Alternative Branch

*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.

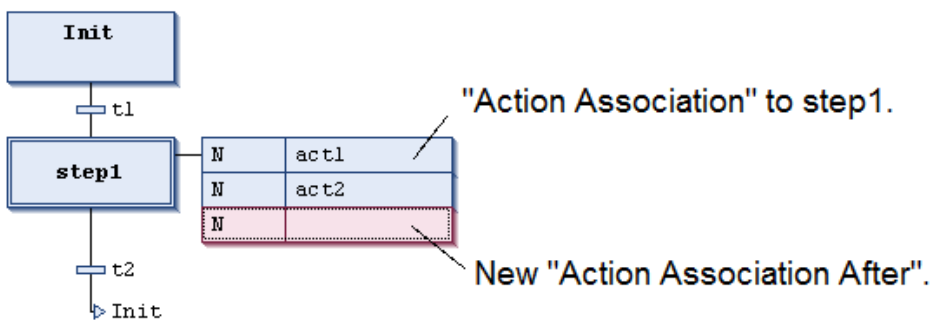
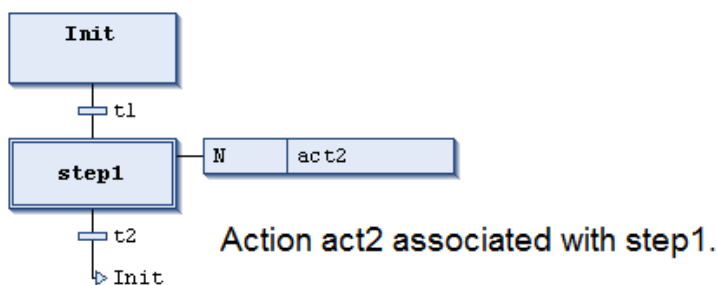


Figure 7-97. Actions Associated to a Step

*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.

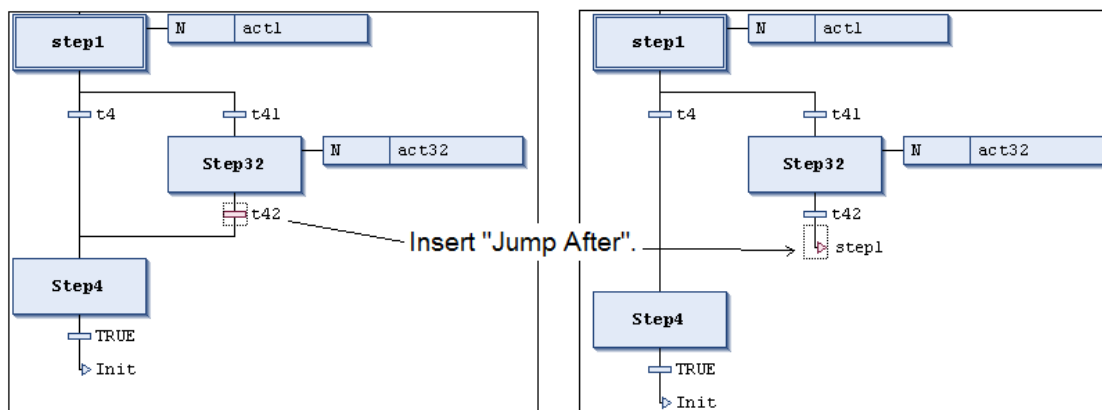


Figure 7-98. Insert Jump After

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*.

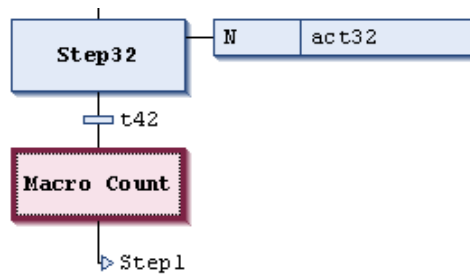


Figure 7-99. Macro Selected in SFC Diagram

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*.

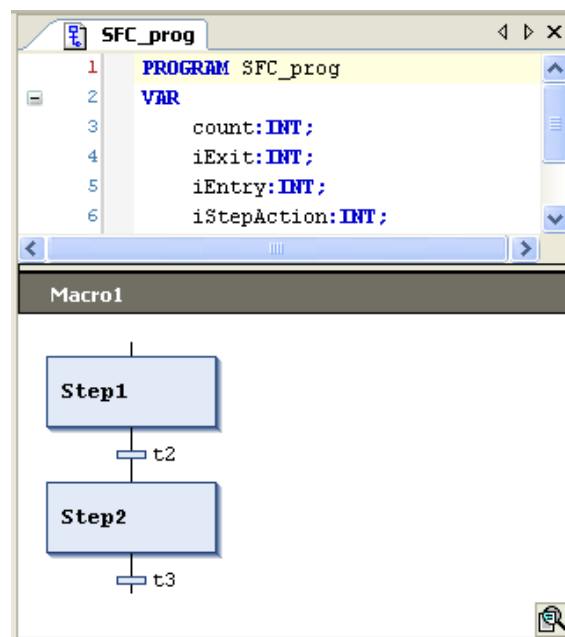


Figure 7-100. Macro Editor

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.

Menu Text List

The *Textlist* menu provides commands for editing a text list. Per default these commands are available in a *Textlist* menu and in the context menu when working in a text list, and the appropriate ones also in the *Visualization* menu.

Available commands:

- Insert Text
- Create Global Text List
- Add Language
- Remove Language
- Import/Export Textlists
- Update Visualization Text IDs
- Check Visualization Text IDs
- Remove Unused Text IDs

Insert Text

Symbol: 

This command of category *Text list* is available in a Textlist menu and in the context menu when working in the Text List editor. It serves for adding a new text in a new line above that one where currently the focus is on. It opens an edit field in the *Standard* column and by default enters the string *NewText*, which you can modify immediately.

Create Global Text List

Symbol: 

This command of category *Text list* can be used to explicitly create a global text list. The `GlobalTextList` object will be added to the POU's view.

`GlobalTextList` is created automatically as soon as the first text is defined in the configuration of a visualization.

Add Language

Symbol: 

This command of category *Text list* is used to add a new language column to a textlist.

Open the textlist, perform the command and enter the language name in the *Choose language* dialog. After confirming with OK the language column will be added rightmost in the current list.

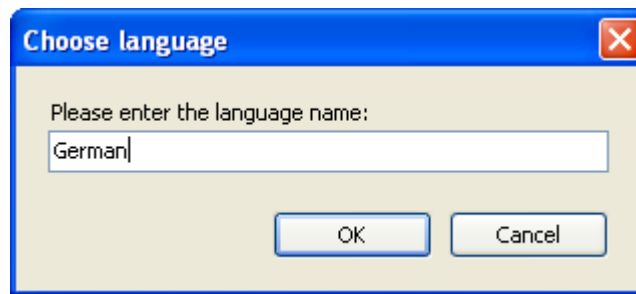


Figure 7-101 Dialog Choose Language

Remove Language

Symbol:

This command of category *Text list* is used to remove that language column from a textlist. Open the textlist, set the cursor in the respective column and perform the command.

Import/Export Textlists

Symbol:

This command of category *Text list* provides the data exchange with other programs as for instance Excel. The data format in use is .csv (Comma Separated Values). The executing the command the following dialog appears:

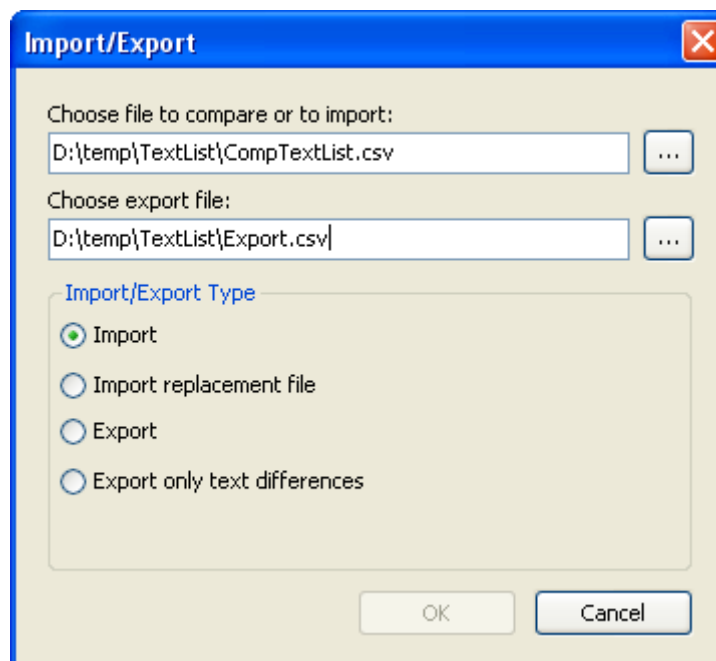


Figure 7-102 Dialog Import/Export Textlists

By entering the corresponding paths or by making use of the input assistant () the files to be imported, exported or compared may be specified. Which of the actions should be executed, may be defined by activating the corresponding item in the lower part of the dialog:

- *Import* : Importing an external file its dataset is put in line with the dataset of the project. The dataset in the project is adjusted according to the following rules:

- If the data content is identical, the data set will be left unchanged.
 - If a translation has been added to the external file, it will be added to the data set of the project as well.
 - If text within a translation has been modified, the modification will be overtaken in the project as well.
 - If translation texts are missing within the external file, the data set in the project will not be modified.
 - If a new line has been added to the external file, this new data record will be incorporated in the data set of the project file.
 - If the project contains an additional line, it will be preserved.
 - A modification within the column *Default* may be considered as insertion of new text. If however there are text positions containing several empty spaces instead of a single one, then this will not be handled as a modification.
- *Import replacement file*: While importing a textlist a modification within column „Default“ is considered as an insertion of a new line. The reason is, that the column „Default“ is serving as key for comparing the lines during import/export. Note: If texts in the column have several blanks instead of one, this is not considered as a change.

If text shall be modified within the column *Default* (elimination of typing error or supplement to existing text), a replacement file becomes necessary.

Example – Import of .csv File

Default old	Default new	Command
Cancel ?	Cancel	REPLACE
Do you want to register ?	Do you really want to register ?	REPLACE_AND_REMOVE
Do you really want to register ?!	Do you really want to register ?	REPLACE_AND_REMOVE

Table 7-3. Importação de Arquivo

The replacement file will be executed top down. Thus, the change history might be accounted for.

The command defines what to do with a text line. The only commands available yet is REPLACE. It will have the following effect:

Normally, the text entered in column *Default* will be replaced by the new text. In the example *Cancel ?* will be replaced by *Cancel* and *Do you want to register ?* by *Do you really want to register ?*. Simultaneously the texts of all visualization elements will be adjusted, i.e. the old text entries within the visualization elements will be replaced.

In the case that the new default text is already contained in the *Default* field of another row of the textlist, the row containing the entry to be replaced will be deleted completely. The visualization elements concerned receive the corresponding entries of the remaining row with the same default entry. In the example this will happen for the default entry *Do you really want to register ?!* that should be replaced by *Do you really want to register ?*. Due to the change history there will already exist a row with this default entry when the related REPLACE command shall be carried out. To avoid multiple occurrences of the key, the row containing the old *Default* text *Do you really want to register ?!* will be deleted completely from the textlist.

- *Export*: Exporting textlists all modifications within the project are compared to an external file. A new export-file will be created according to the following rules:
 1. If the data content is identical, the data set will be exported as it is.
 2. If a translation has been added to the project file, it will be included.
 3. If text within a translation has been modified, the modification in the project will be overtaken in the export file as well.
 4. If translation texts are missing within the project file, the translations of the template will be used for the new data set.

5. If a new line has been added to the project, this new data record will be incorporated as new data set of the project file.
 6. If the external file contains an additional line, it will be exported again.
 7. A modification within the column Default may be considered as insertion of new text.
- *Export only text differences:* If this option is activated, only the lines differing from their corresponding line in the different versions are included in the export file. Such a difference file is suited as input for translation purposes. As the file is intended to be kept as small as possible, missing items in the actual textlists will not be treated as differences.

ATTENTION:

For locating the corresponding data sets the column Default is used for the GlobalTextList and the column Id for all other textlists. Therefore the column entitled Id is empty for all data sets of the GlobalTextList.

Example – Importing .csv File

Data content of external file:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		
TextList1	2	Light		

Table 7-4. Data of External File

Data content of textlist of project before import:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abortion
TextList1	1	Door	Tür	Door
TextList2	3	Seat	Sitz	Seat

Table 7-5. Text List before Import

During the import all differences are incorporated into the project. Thereby the two lists are adapted so that the following textlist will result in the project.

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door	Tür	Door
TextList1	2	Light		
TextList2	3	Seat	Sitz	Seat

Table 7-6. Text List Resultant

Example - Export of a .csv File

Data set of external file:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abort
TextList1	1	Door	Tür	Door
TextList2	2	Seat	Sitz	Seat

Table 7-7. Data of External File

Data content of textlists of project before export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering whee
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		
TextList1	3	Light		
TextList2				

Table 7-8. Data before Export

During the export all differences are incorporated in the external file. Thereby the two lists are adapted so that the following external file will be created.

Data content of textlists of project after export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door	Tür	Door
TextList1	3	Light		
TextList2	2	Seat	Sitz	Seat

Table 7-9. Data after Export

Example - Export of Text Differences Only

Data content of textlists of project before export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel		
TextList1	0	Cancel	Abbrechen	Abort
TextList1	1	Door	Tür	Door
TextList2	2	Seat	Sitz	Seat

Table 7-10. Data of External File

Data content of textlists of project before export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Automobile	Automobil	Automobile
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	1	Door		

TextList1	3	Light		
TextList2				

Table 7-11. Data before Export

During the export all lines differing from the corresponding ones (line 2,3 and 5 of the actual list) are included in the export file.

Data content of external file after export:

Text List	ID	Default	Deutsch	English
GlobalTextList		Steering wheel	Lenkrad	Steering wheel
TextList1	0	Cancel	Abbrechen	Cancel
TextList1	3	Light		

Table 7-12. Dados after Export

Update Visualization Text IDs

Symbol: 

This command belongs to category *Text List*. If a static text is modified within a visualization element, the visualization and eventually the GlobalTextList as well must have write permission. If modifications will be done, though the write permission is missing, it can be, that the text-ids do no longer fit to the texts of a visualization element.

By use of the command *Update Visualization-Text -IDs* these errors can be corrected automatically. Therefore all affected visualizations and the GlobalTextList must have write permission.

Check Visualization Text IDs

Symbol: 

This command belongs to category *Text List*. If a static text is modified within a visualization element, the visualization and eventually the GlobalTextList as well must have write permission. If modifications will be done, though the write permission is missing, it can be, that the text-ids do no longer fit to the texts of a visualization element.

By use of the command *Check Visualization-Text-IDs* such errors can be detected in the visualizations.

Remove Visualization Text IDs

Symbol: 

This command of category *Text list* serves to delete texts being no longer used in a visualization element from the GlobalTextList.

Menu Visualization

The *Visualization* menu provides commands for editing a display object in the view editor.

Available commands:

- Interface Editor
- Hotkeys Configuration
- Element List
- Activate Keyboard Usage
- Alignment

- Order
- Group
- Ungroup
- Background

Interface Editor

Symbol: 

Default Shortcut: <ALT>+<F6>

This command (category *Visual Commands*) opens the *Basics of Interface Editor* for defining frame parameters in a visualization which is intended to get referenced in a **Frame** element in another visualization.

Hotkeys Configuration

Symbol: 

This command (category *Visual Commands*) opens the *Editor for Hotkeys Configuration* for the current visualization. It will be displayed in a tabbed view in the upper part of the visualization editor.

Element List

Symbol: 

This command (category *Visual Commands*) opens the **Element List** of the current visualization. It will be displayed in a tabbed view in the upper part of the visualization editor

Activate Keyboard Usage

Symbol: 

This command (category *Visual Commands*) is available in the menu bar for an integrated visualization (**Diagnosis Visualization**). It activates resp. deactivates the **Keyboard Usage in Online Mode**.

When the keyboard operation is activated, any inputs on elements and the selection of elements can be done by using certain keys. In this case other commands given via key shortcuts will not be executed as long as the visualization editor is active and in online mode.

Alignment

Align Left

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the left line of that of those elements which is at the left- most position.

Align Top

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the upper line of those elements which is at the upmost position.

Align Right

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the right line of that of those elements which is at the right- most position.

Align Bottom

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the bottom line of that of those elements which are at the most bottom position.

Align Vertical Center

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the vertical center of all selected elements.

Align Horizontal Center

Symbol: 

Using this command (category *Visual Commands*) currently selected visualization elements will be aligned to the horizontal center of all selected elements.

Order

Bring One to Front

Symbol: 

This command (category *Visual Commands*) places the selected element one layer higher, that means nearer to the foreground of the visualization.

Bring to Front

Symbol: 

This command (category *Visual Commands*) places the selected element in the absolute foreground of the visualization.

Send One to Back

Symbol: 

This command (category *Visual Commands*) places the selected elements one layer deeper that means nearer to the background of the visualization.

Send to Back

Symbol: 

This command (category *Visual Commands*) places the selected element in the absolute background of the visualization.

Group

Symbol: 

This command (category *Visual Commands*) groups the currently selected visualization elements and displays the group as a single selected object. For multiple selection keep the [Shift] key pressed while clicking on the desired elements. Alternatively you might perform a mouse-click outside of an element in the editor window and - while keeping the mouse-button pressed - draw a rectangle around the desired elements. For resolving the group use command **Ungroup**.

The following picture shows the grouping (from left to right) and ungrouping (from right to left) of two rectangle elements:

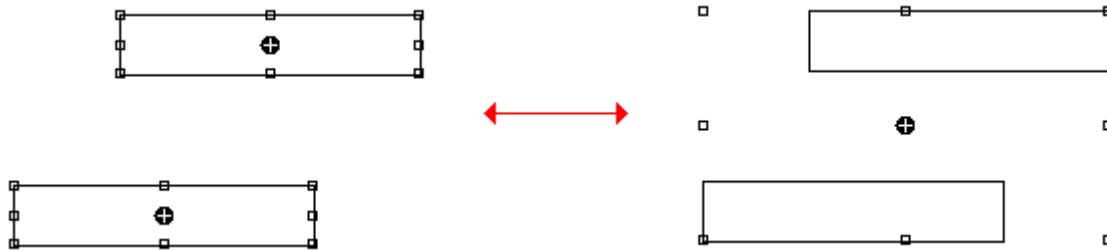


Figure 7-103. Agrupando e desagrupando

Ungroup

Symbol: 

This command (category *Visual Commands*) resolves a selected group of elements. The particular elements will be displayed each selected. See also **Group**.

Background

Symbol: 

This command (category *Visual Commands*) allows to define a image filling the background of the complete visualization. The dialog *Background* will be opened, where you can define an image file and/or an background color. For this purpose activate the respective option.

Bitmap: To define a background image, you must enter the path of an image file, which is available in an image pool, in the Bitmap field: Enter the name of the image pool and the ID string separated by a dot: <pool name>.<image ID> (for example "Images_1.drive_icon", "Images_1.43").

Color: To define a background Color, choose one from the available color selection list.

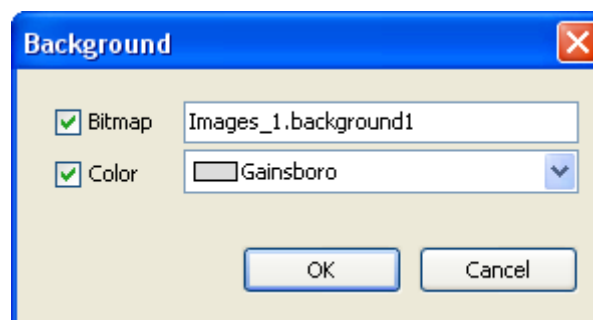


Figure 7-104. Background Dialog

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 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.

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
- Redundancy Configuration
- OPC Configuration
- CPU Information
- Reset warm *
- Reset cold *
- Reset origin*
- Simulation
- Export Online Variables
- Import Online Variables

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

This command is only available in online mode when logged with an online user. It performs a logoff of the online user. For further information, check **User and Access Right Management**.

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 affect the desired behavior of the system.

Depending on the controlled system, damages to machines and parts could result, or even health and life of persons could be endangered.

NOTES:

- When an online change is done, the application-specific initializations (homing etc.) will not be executed because the machine keeps its state. For this reason, the new program code might not be able to work as desired.

- Pointer variables keep their values form the last cycle. If there is a pointer on a variable, which has changed its size due to an online change, the value will not be correct any longer. Make sure that pointer variables get re-assigned in each cycle.

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*.

Redundancy Configuration

This command allows the user to configure whether the CPU connected will be configured as PLC A, PLC B or Non-Redundant.

You can also configure whether there will be project synchronization between the PLCs, if the redundancy of PLCs is being used.

For further information consult the manual of the corresponding CPU.

OPC Configuration

Through this command it's possible to configure the OPC Server installed with the MasterTool IEC XE. For further information about the OPC Server configurations check the **OPC Communication Editors** section.

CPU Informations

This command shows a screen with information about the CPU. To access this option it's necessary to select a device as described in the **Communication Settings** section. If this operation wasn't performed previously, an error message will be displayed. Figure 7-105 shows the *CPU Informations* screen.

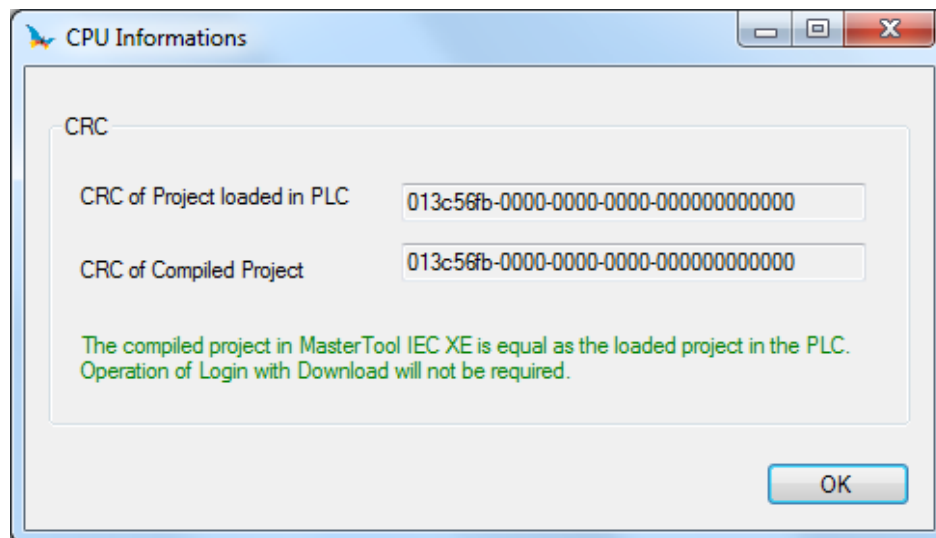


Figure 7-105. CPU Informations

CRC

Inside the CPU informations screen, there's the CRC of the Project loaded in the PLC and the CRC of the Project compiled in the MasterTool IEC XE. By creating a new project without compiling it, the value shown for the CRC of Compiled Project will be always reset. After compiling the project for the first time the CRC value will be shown.

Similarly, if there's no project loaded in the PLC, the value of the CRC of Project Loaded in PLC is going to be reset. By loading a program for the first time the value is going to be shown. If the value from both fields are equal, a message will be shown, informing that the project compiled and the project loaded in the PLC are equal and the Login command can be executed without the need to download. If there are difference between the two fields, it will be informed that it's necessary to perform a download in the PLC since the projects are different.

NOTES:

- The *Reset Origin* command in the Online menu will reset the CRC of the Project loaded in PLC.
- The Clean All command in the Compile menu will reset the CRC of the Compile Project.
- When the two CRC fields are reset, it will be indicated that the projects are equal, even if it's necessary to perform a download to load the correct application into the PLC.

ATTENTION:

The CRC of the Project shown in this screen is different from the CRC of the application shown in the Nexto Series CPU diagnostics.

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 7-106 for an example: Login has just been performed for the currently active application.

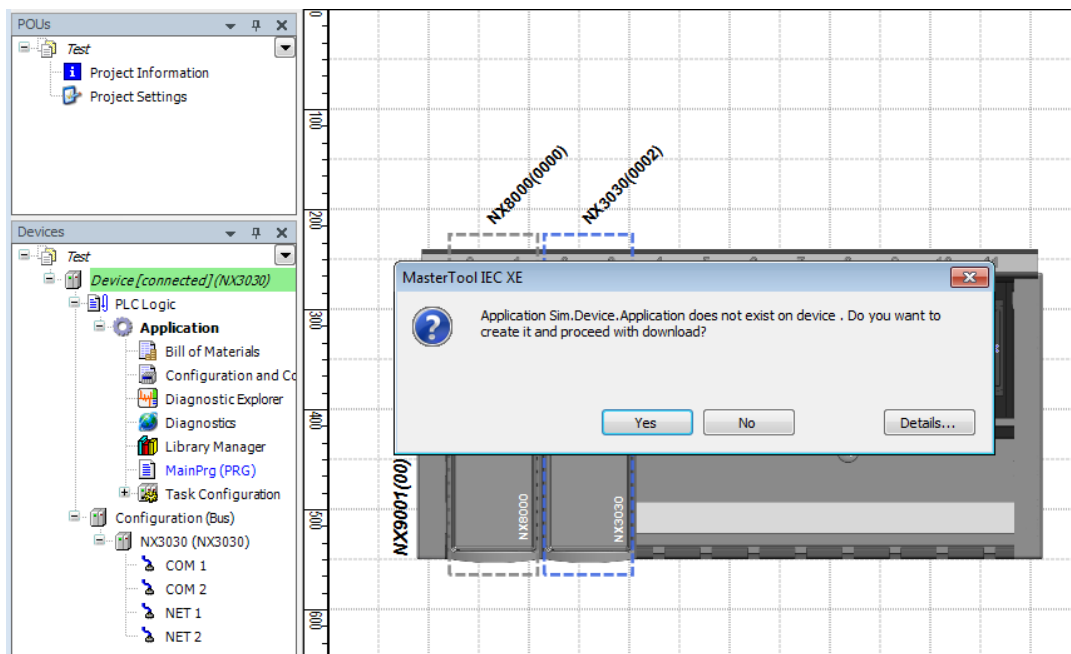


Figure 7-106. Login to the Simulator Target

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**.

Export Online Variables

The Export Online Variables command allows to export the values of the variables declared in POU's and GVL's to a file named X_OnlineVarsRef, where X is the name of the MasterTool IEC XE project. This feature allows the saving of the application configuration state in a given time, as it allows its restoration to that state using the Import Online Variables command.

This is extremely useful when the adjustments and configuration values are not stored in an HMI or a SCADA System with which the application is operated. In cases of application failure, CPU failure,

or any other hardware element that causes loss of information, the created file can be used to restore the values after loading the project into the PLC again.

The file is going to be created in the directory where the project is opened. By running the Export Online Variables command, the screen in Figure 7-107 will be displayed.

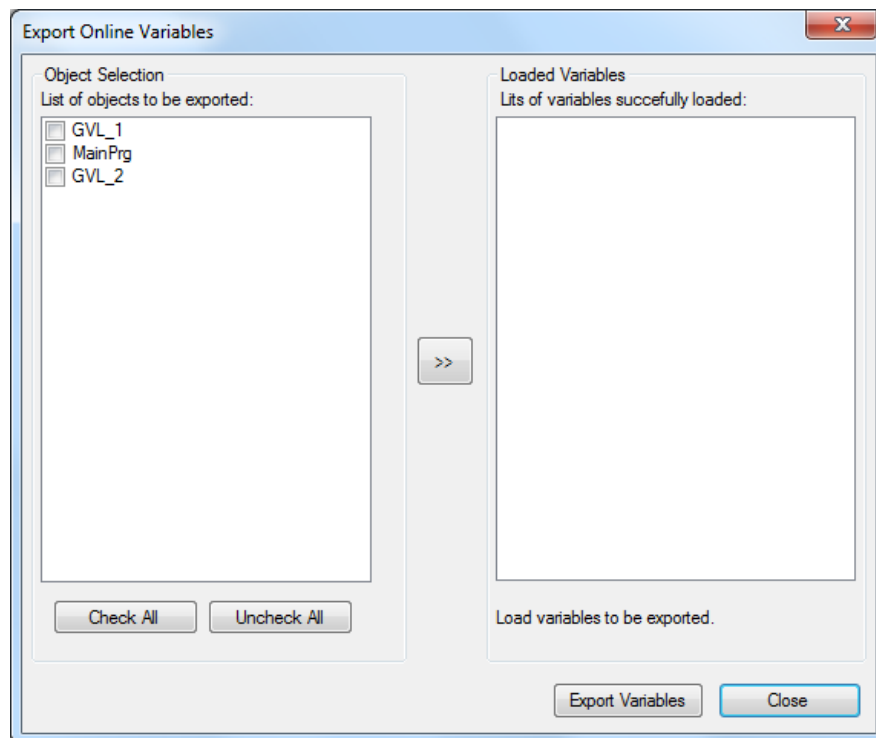


Figure 7-107. Export Online Variables Screen

In the left side of the screen it will be shown all POU's and GVL's available in the project, allowing the selection of which will have its values read and saved in a file. It's possible to configure individually all objects that are going to have its value exported or, to ease the editing process, it's also possible to use the *Check All* and *Uncheck All* buttons.

After selecting the objects to be exported and press the >> button, in the center of the screen, all the variables content and its respective values will be stored by the MasterTool IEC XE in internal structures of the tool. The screen in Figure 7-108 will be shown and all variables present in the selected objects will be at the right side of the screen.

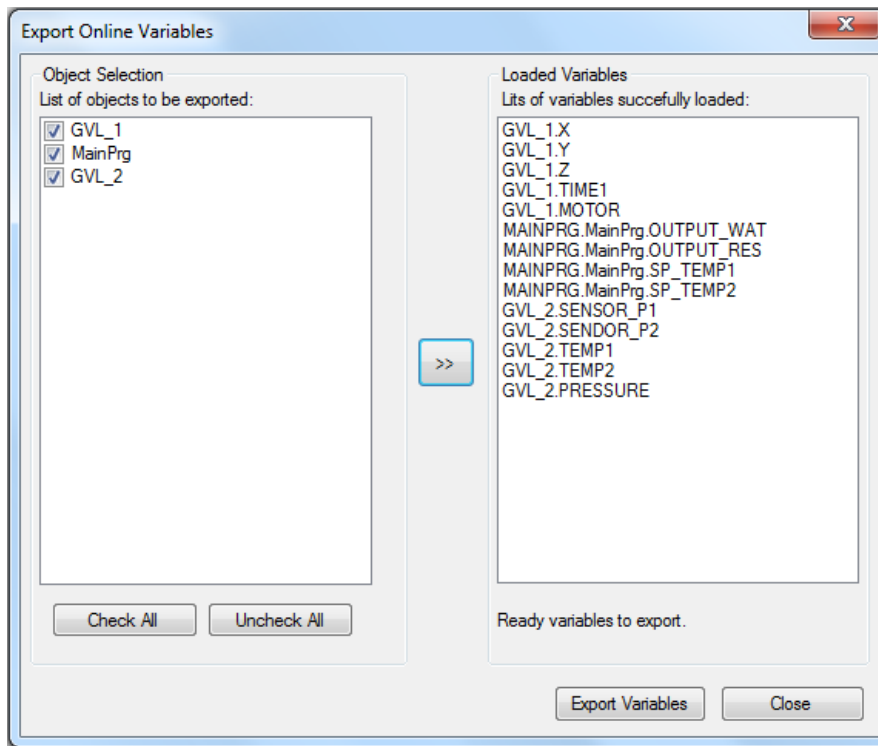


Figure 7-108. Variables to be Exported by the Export Online Variables Command

By pressing the Export Online Variables button, the content stored by the tool is saved in the file created in the directory of the project. If the file already exists, it will be modified and will have only the content of the last export action performed.

This command is only available after a command to Login into a CPU is done, that is, it's necessary to be Online for the command to appear in the menu. The same monitoring mechanism that is used to monitor variables in GVLs, POU's and Monitoring Windows is also used to load the data to be saved by the command. This mechanism has a limit of data that can be read by an open instance of the MasterTool IEC XE. The limit is 30,000 elements.

Because of that, it's not possible to save more than 30,000 elements in a file created by this command. By selecting a quantity of POU's and GVL's in which the sum of elements is larger than 30,000, MasterTool IEC XE will inform that the limit will be exceeded and the operation won't be made.

In the case of simple variables, each variable represents an element. In the case of types such as Structs and Arrays, each element represent a read value that is going to be counted in this limit. Beyond this limitation on the maximum quantity of elements, the Array data type is limited by maximum quantity of bytes. Arrays larger than 30,000 bytes can't be exported. This limitation is verified by the tool.

The export and import of large volumes of data (near the 30,000 limit) consumes shared resources with MasterTool IEC XE's monitoring tool. These resources may deplete throughout the program's execution. In such case, the restoration of these resources is only achieved by restarting the program.

ATTENTION:

Keeping in mind the limit of variables that can be exported, it is important to define a strategy for declaring them. This way, it will be easier to select variables that really matter and need to be saved. Variables that define adjustments or application configurations must be saved to enable its restore. Variables cyclically recalculated by the PLC logic don't need their values saved.

ATTENTION:

This feature enables operations in Simulation mode. However, the limits of this operation in such mode are smaller than they are in an actual PLC – that is, it's not possible to perform operations with 30,000 elements, and MasterTool IEC XE won't consist it.

A good practice to store values of a given project is to create a Project Archive. This way, saved variables can be stored along with application backups. To create a Project Archive, go on File/Project Archive/Save/Send Archive. Figure 7-109 shows the Project Archive screen.

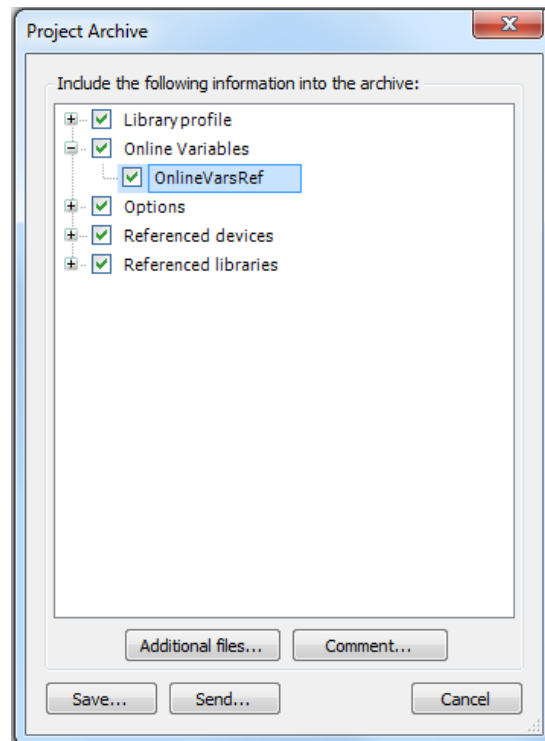


Figure 7-109. Online Variables in Project Archive

If the Online Variables option is marked, once the Project Archive is created, it can be extracted and the saved values can be extracted along with the application.

Import Online Variables

The Import Online Variables command enables the importation of variable values previously exported through the Export Online Variables command. The file to be imported must be in the same folder of the project file, and its name must be X_OnlineVarsRef, where X is the name of the project. This command is only available after logging into the CPU, that means it's necessary to be Online for the command to be enabled.

To enable this operation, the file must either be created in the project's work folder or extracted from a Project Archive at the File/Project Archive/Extract Archive menu. The extracted project must have been created with the Online Variables option marked.

If the project is edited before importing, some inconsistencies may have been created between the exported file and the project. Below is a list of such situations and the behavior in each case.

- Variable present in the exported file, but removed from the project: the removed variable won't be written and MasterTool IEC XE will inform which variables couldn't be imported by the end of the process.

- Variable added to the project, but absent from the exported file: the added variable will keep its value after importing.
- Exported variable's type is different from the project variable: the variable whose type has been altered won't be written and MasterTool IEC XE will inform which variables couldn't be imported by the end of the process.
- String type variable has its size modified: if the string in the project is equal or larger than the one exported, it will be imported; otherwise, it won't be written and MasterTool IEC XE will inform which variables couldn't be imported by the end of the process.
- Struct Element or Array Position present in the exported file, but removed from the project: the removed variable won't be written and MasterTool IEC XE will inform which variables couldn't be imported by the end of the process.
- Struct Element or Array Position added to the project but absent from the exported file: the added variable will keep its value after importing.

If there are no restrictions to the import, MasterTool IEC XE will display a message when the process is done successfully. If there are any restrictions, a window with all variables that were not imported can be consulted by the end of the process.

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 Watch list
- 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).

NOTES:

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.

- Breakpoint usage in tasks with priority below 5 may cause an user application watchdog.

New Breakpoint

Symbol: 

This command is used to insert a new breakpoint in one of the POUs. It does not care where the cursor currently is placed, the *New Breakpoint* dialog will open where in sub-dialog *Location* you can choose one of the possible breakpoint positions all over the project and in sub-dialog *Condition* can define some conditions for the new breakpoint.

NOTE: For setting a breakpoint at the current cursor position notice command **Toggle Breakpoint**.

Location

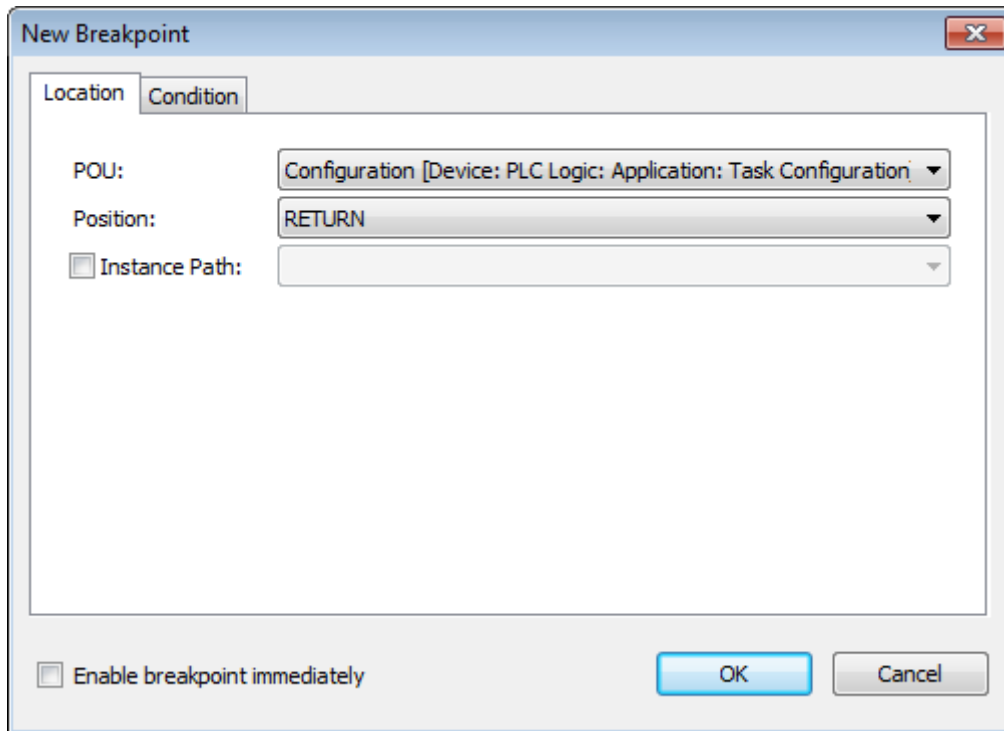


Figure 7-110. New Breakpoint Dialog, Location

- *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

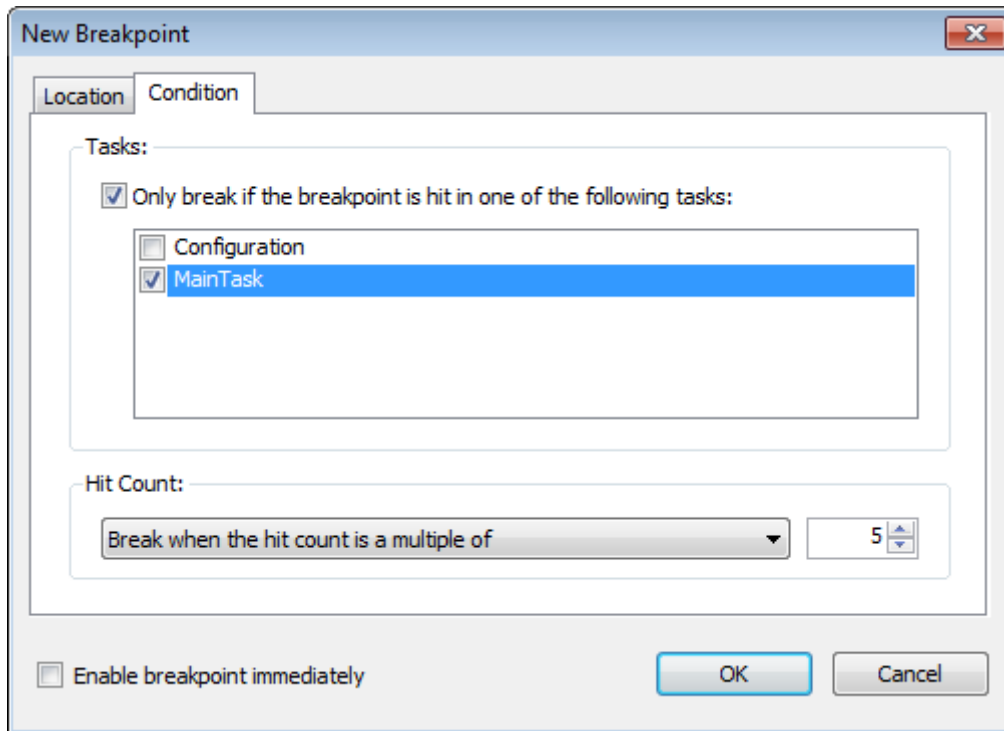


Figure 7-111. New Breakpoint Dialog, Condition

- *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 7-112, Figure 7-113 and Figure 7-114 show the possible symbols assumed by breakpoints.

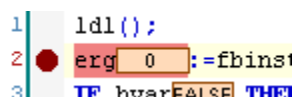


Figure 7-112. Breakpoint in Online Mode

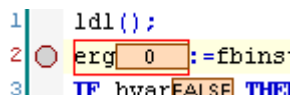


Figure 7-113. Disabled Breakpoint

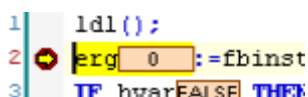


Figure 7-114. Program Stop at Breakpoint

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 7-115. 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 (watch list);
- In the online view of the object within the declaration part of the respective editor.

NOTE: See in this context also the **Force Values** command for permanently setting a defined value.

Example:

Open an object in online mode, for example a program written in ST. In the declaration part of the editor window you will find the displayable expressions listed in a table. Click on the concerned field in column *Prepared Value*, enter the desired value. Perform command *Write Values* that is by default part of the Online menu. The value will immediately be written to the respective field in column *Value*, which means that it is written to the controller. The Prepared value field will be empty again.

The same can be processed in a watch view containing the desired expression.

Notice in this context the *Prepare Value* dialog, which is available for currently forced variables and where a new value to be written can be defined.

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 7-116 presents the sequence of processing in a cycle of application program.

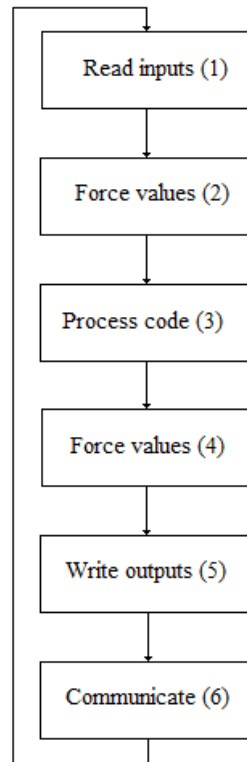


Figure 7-116. Sequence of processing in a cycle

If a forced variable has its value changed during the code execution (3), its value will keep changed until the follow force values process. This kind of situation can generate some inconsistencies during the logic monitoring. It happens because the monitored value can be different of the real one used by the application program, since the communication is treated only in the end of CPU sequence of processing.

NOTE: See in this context also the **Write Values** command for setting a defined value only once at the beginning of a cycle.

The forcing will remain active until it is explicitly suspended by the user for particular and for all variables, or until the application gets logged-out.

ATTENTION:

The forcing operation does not actuate in %I or %Q operands updated with the functions REFRESH_INPUT and REFRESH_OUTPUT. These functions execute a read to %I operands or a written from %Q operands after executing them and they do not consider the forcing effects. For this reason, it is not recommended forcing operand that have been updated by the functions REFRESH_INPUT and REFRESH_OUTPUT that are active in the application program.

To prepare variables for forcing, the desired value must be defined in online mode in one of the following places, which are used for monitoring:

- In a watch view defined in the project, containing a list of variables to be monitored (watch list).
- In the online view of the object within the declaration part of the respective editor.

- In the online view of the object within the implementation part of the FBD /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 7-117. Forced Values in the Declaration Editor of a POU (Online View)

Prepare Value Dialog

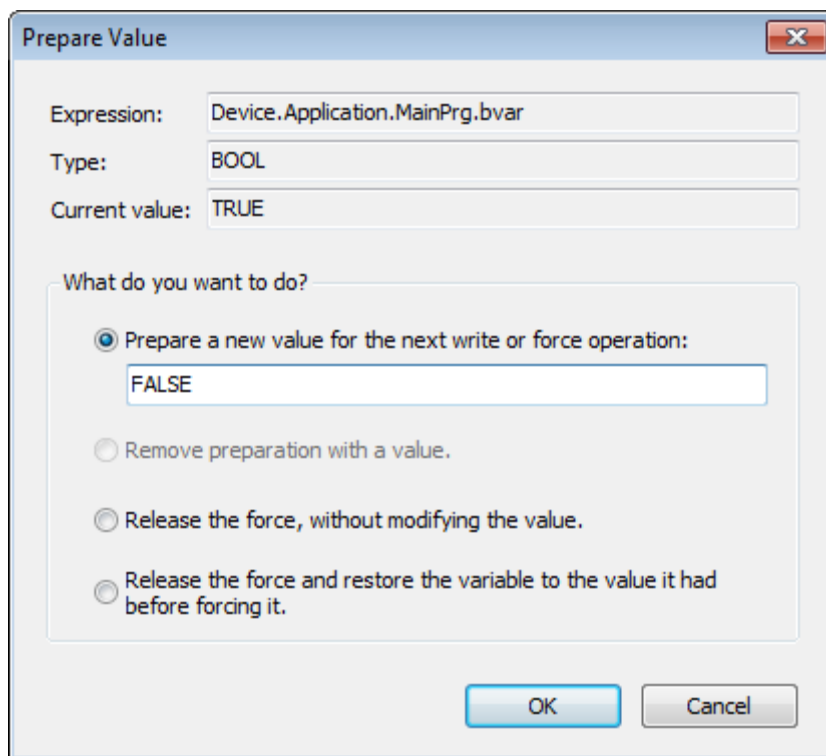


Figure 7-118. Prepare Value Dialog

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 7-119. 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 *STRING*s, 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 Watch List

This command is available in online mode when one of the watch views of watch 1, 2, 3 or 4 is active. It serves to add all currently prepared or already forced variables of the active application to this watch list. Notice however that this works only for docked watch views.

Notice also the possibility to use the watch view *Watch all Forces*, which automatically contains all currently prepared or forced values and additionally provides commands for releasing the forces.

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.

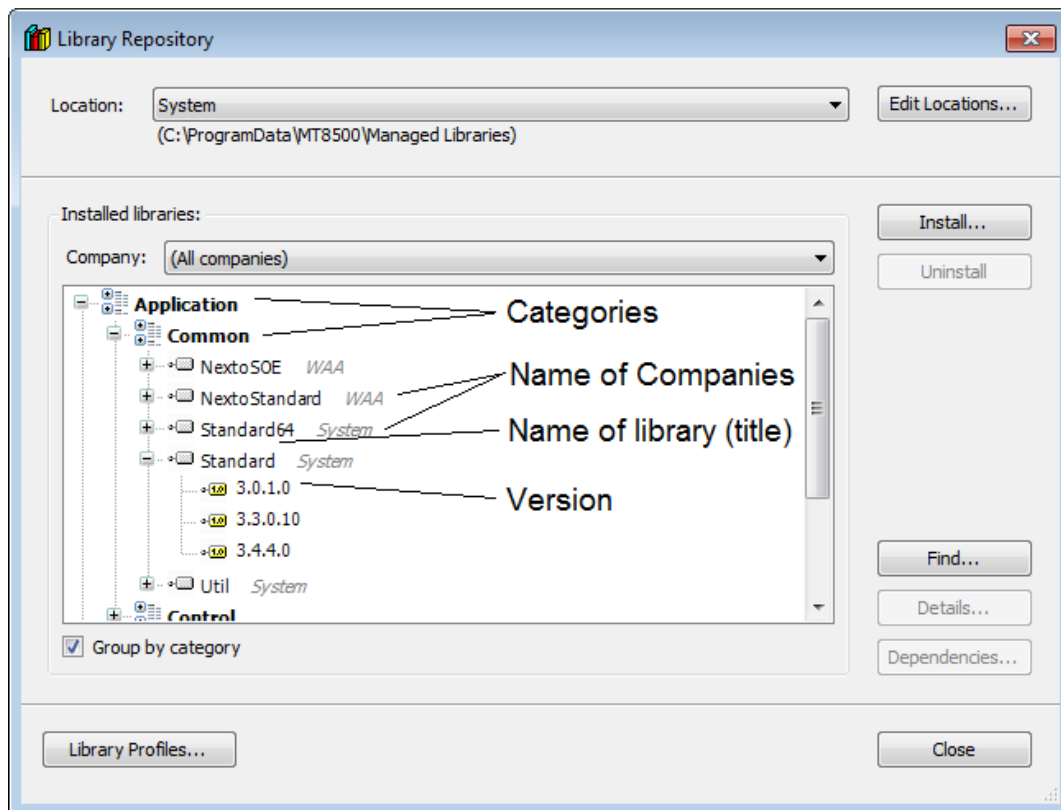


Figure 7-120. Library Repository

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.

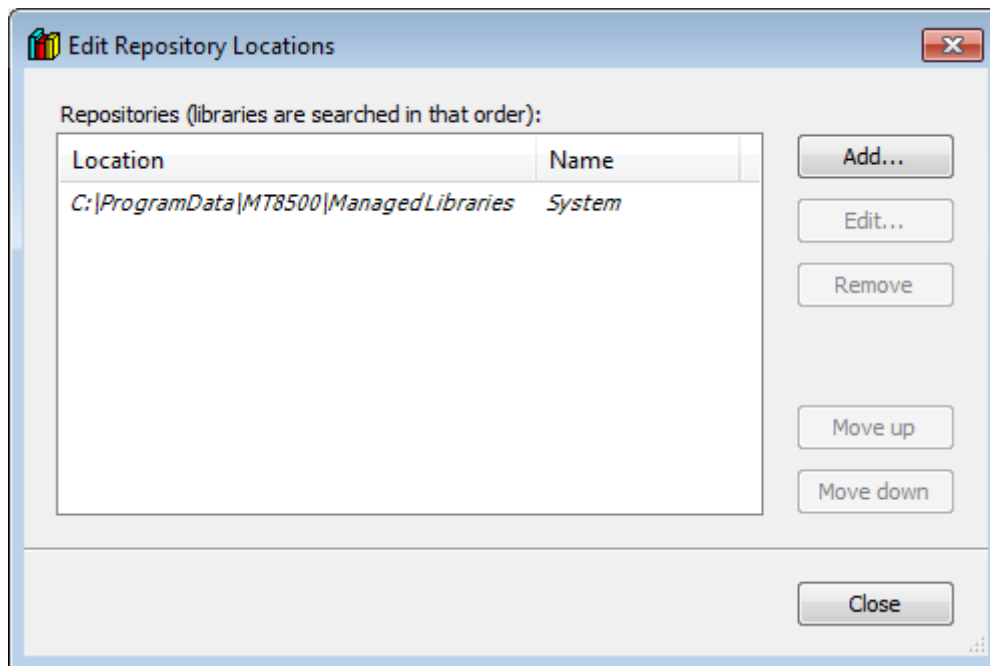



Figure 7-121. Edit Repository Locations

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.

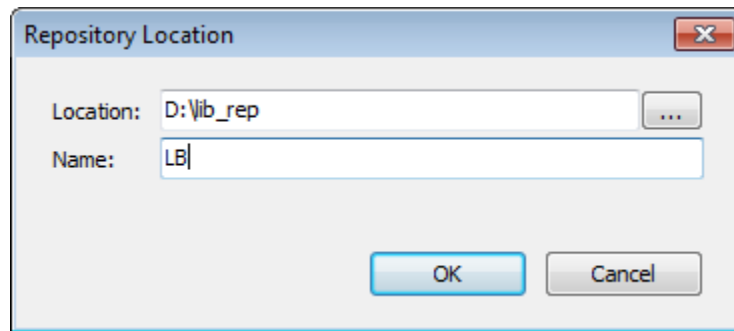


Figure 7-122. Dialog Edit Repository Locations, Add or Edit a Repository

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.

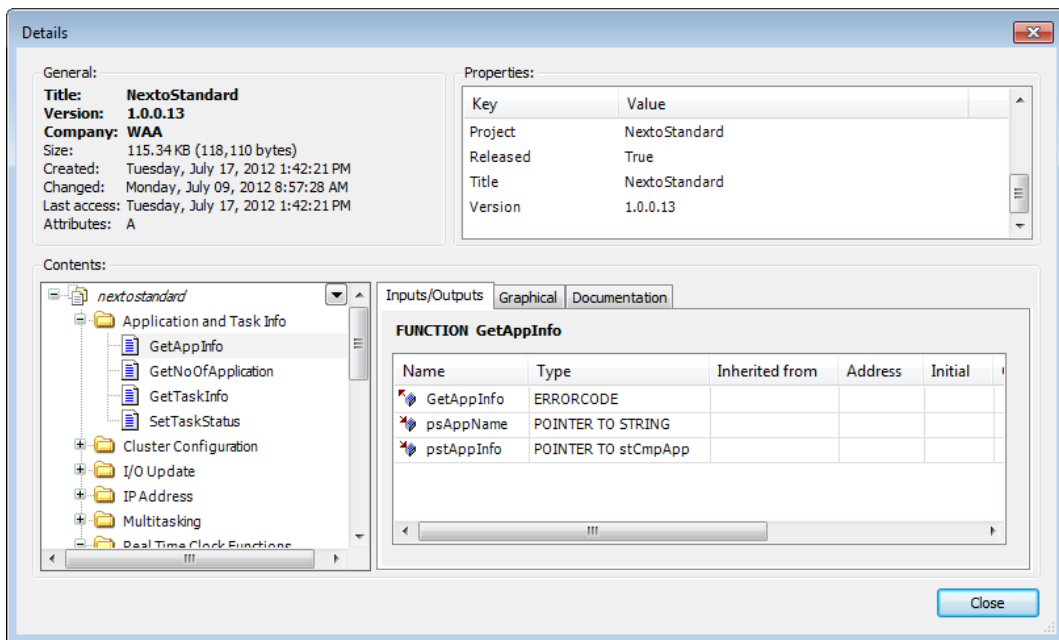


Figure 7-123. Details

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

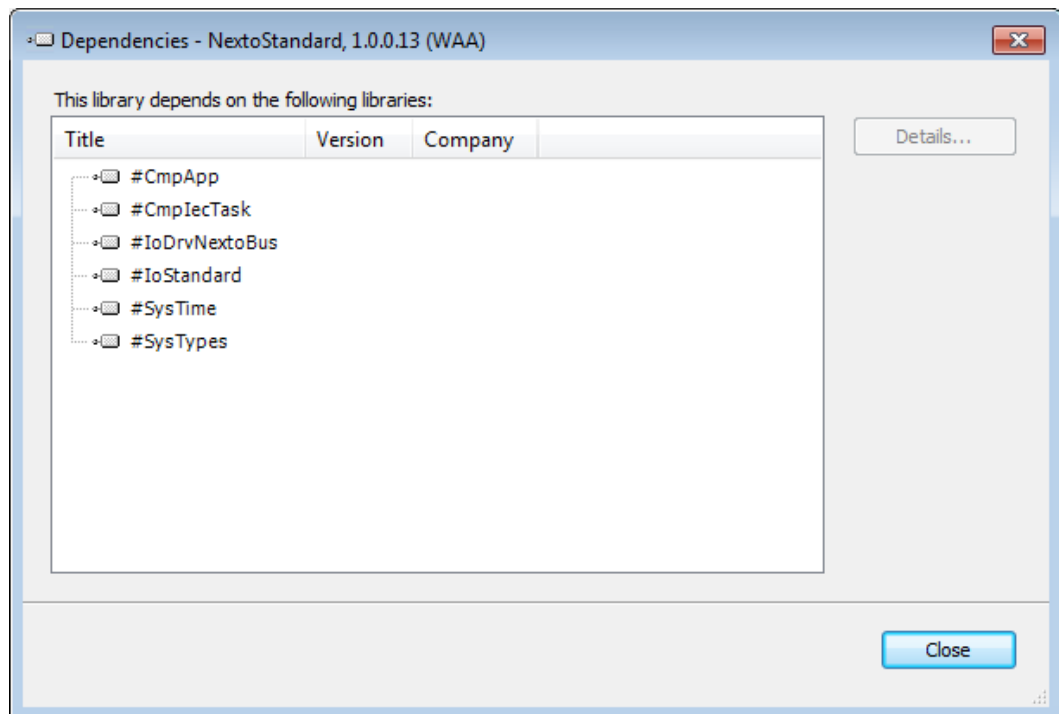


Figure 7-124. Dependencies

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**.

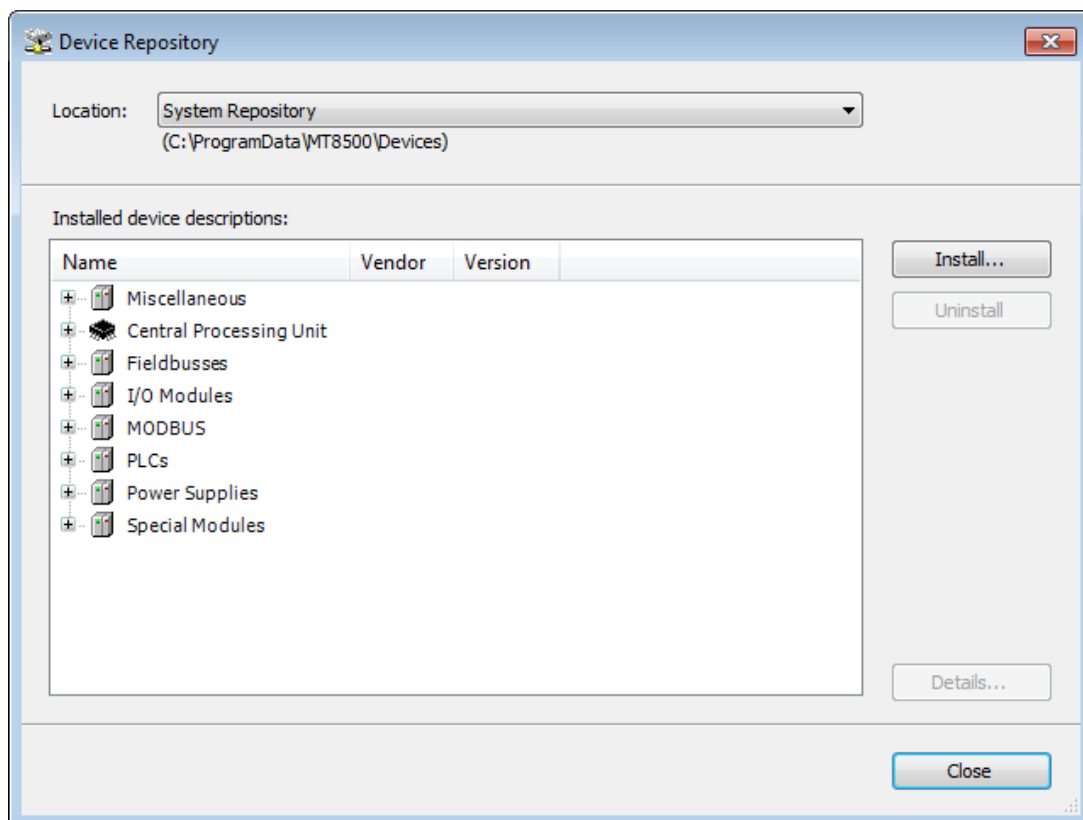


Figure 7-125. Device Repository

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 7-125). 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
- Visualization
- Visualization do Gerenciamento do Usuário

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.

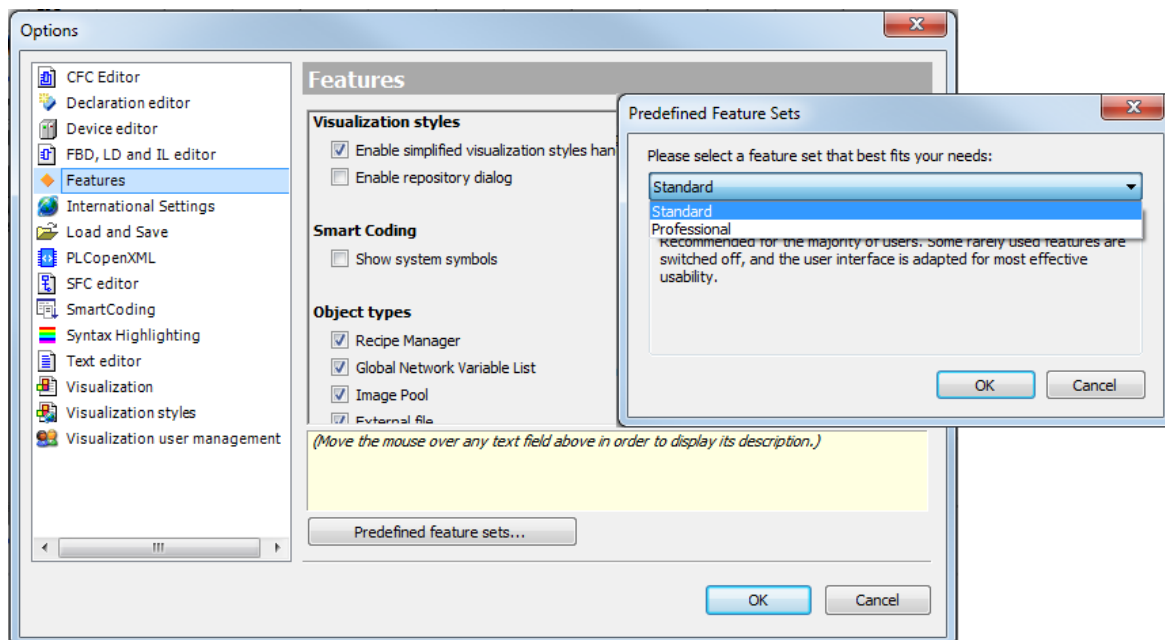


Figure 7-126. Options Dialog, Features

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	<p>Only if this feature is activated, the following object types will appear in the "Add object" context menu:</p> <ul style="list-style-type: none"> - Method - Property <p>Furthermore, when creating a new POU, the wizard will offer to extend FBs or implement interfaces.</p> <p>The new tabular declaration editor also will offer to extend FBs, implement interfaces, and decorate the entire POU or single declarations with compiler pragmas.</p> <p>The input assistant will offer declaration keywords that are used together with object oriented programming (EXTENDS, etc..).</p>	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	<p>If activated, the following simplifications will apply:</p> <ul style="list-style-type: none"> - 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 	On	Off

	<p>references.</p> <ul style="list-style-type: none"> - 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. 		
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 be displayed so that the user can remove the reference manually.	On	Off
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 7-13. 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.

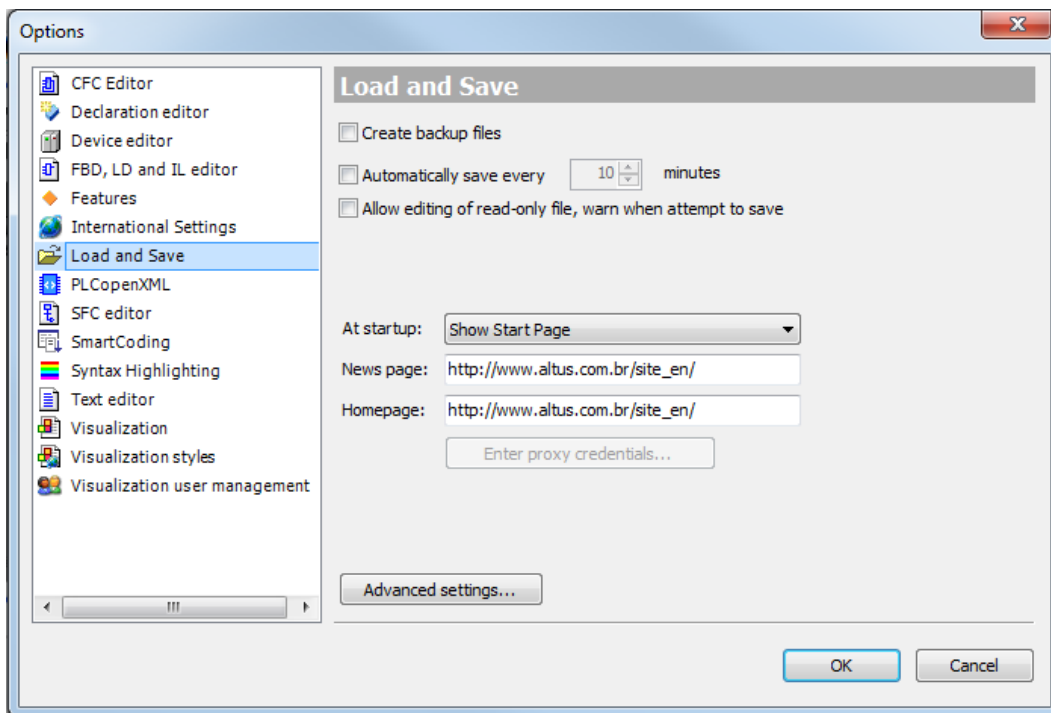


Figure 7-127. Load and Save Options Dialog

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.

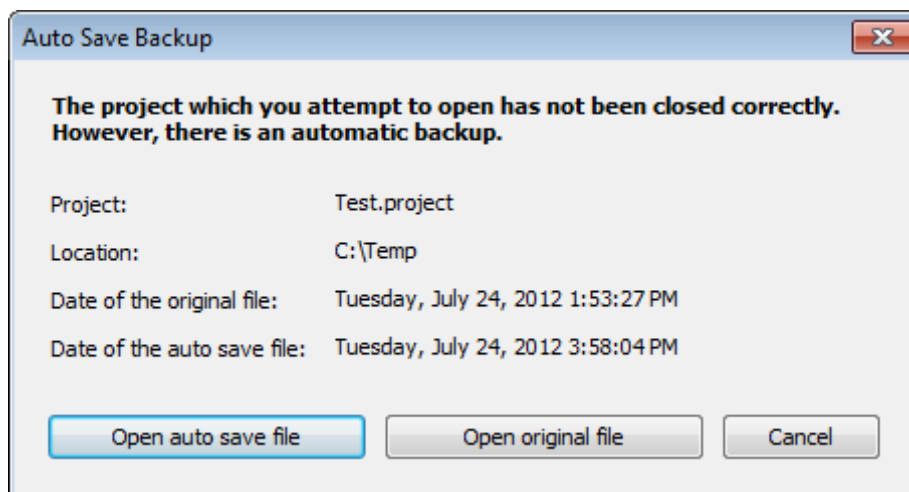


Figure 7-128. Auto Save Backup Dialog

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.

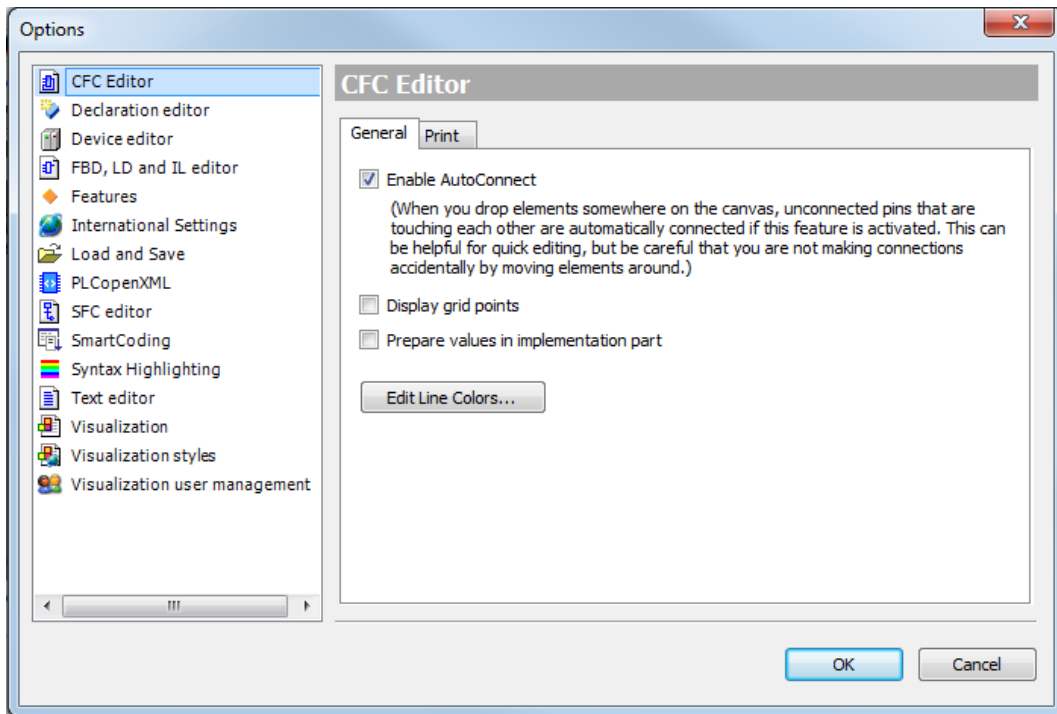


Figure 7-129. Enable AutoConnect

Declaration Editor

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

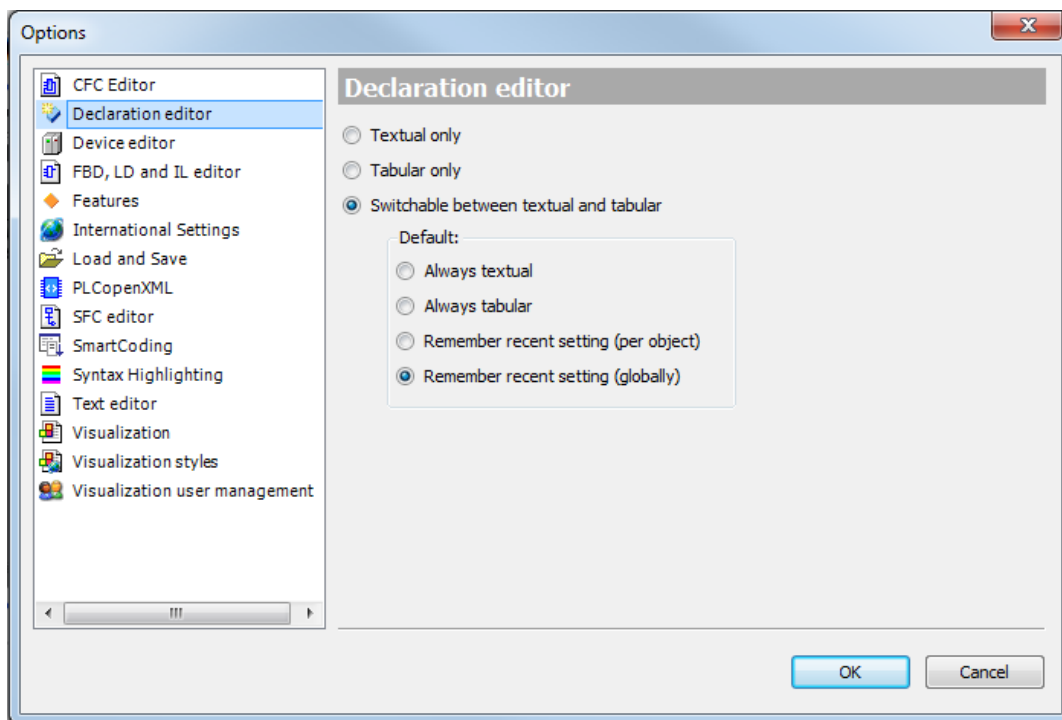


Figure 7-130. Declaration Editor Options

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

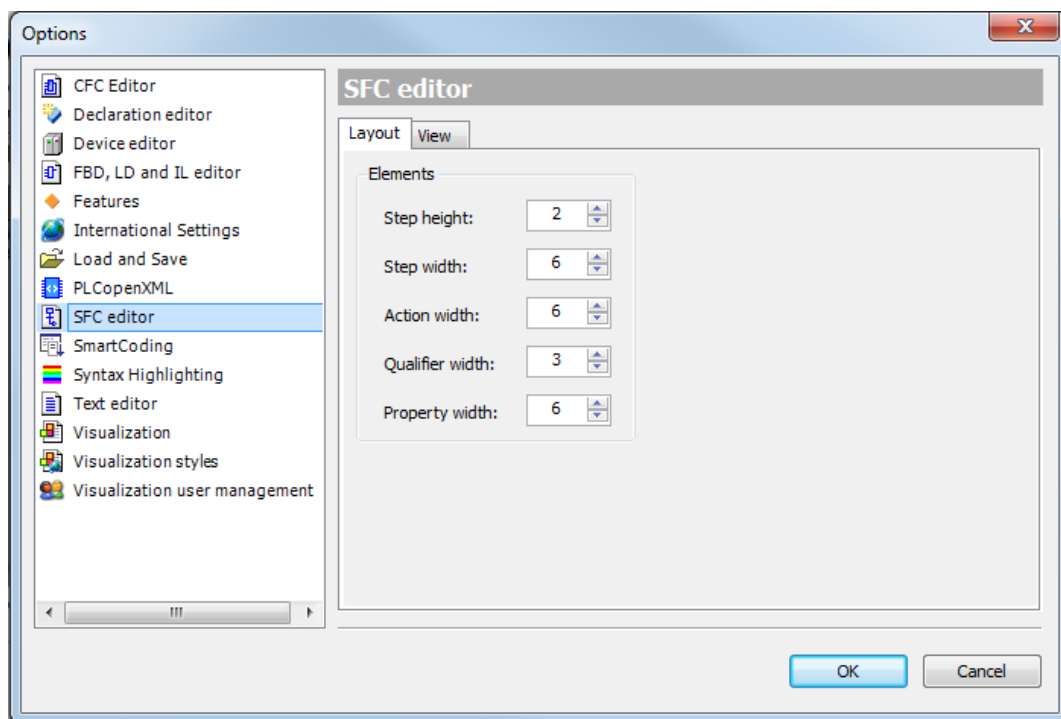


Figure 7-131. SFC Editor Options Dialog, Category Layout

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

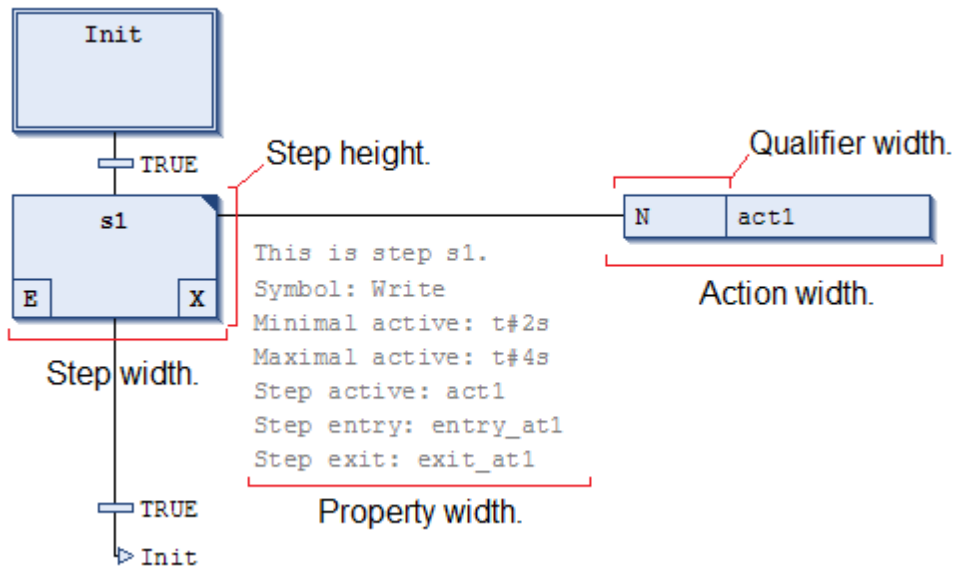


Figure 7-132. Layout Options

- *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

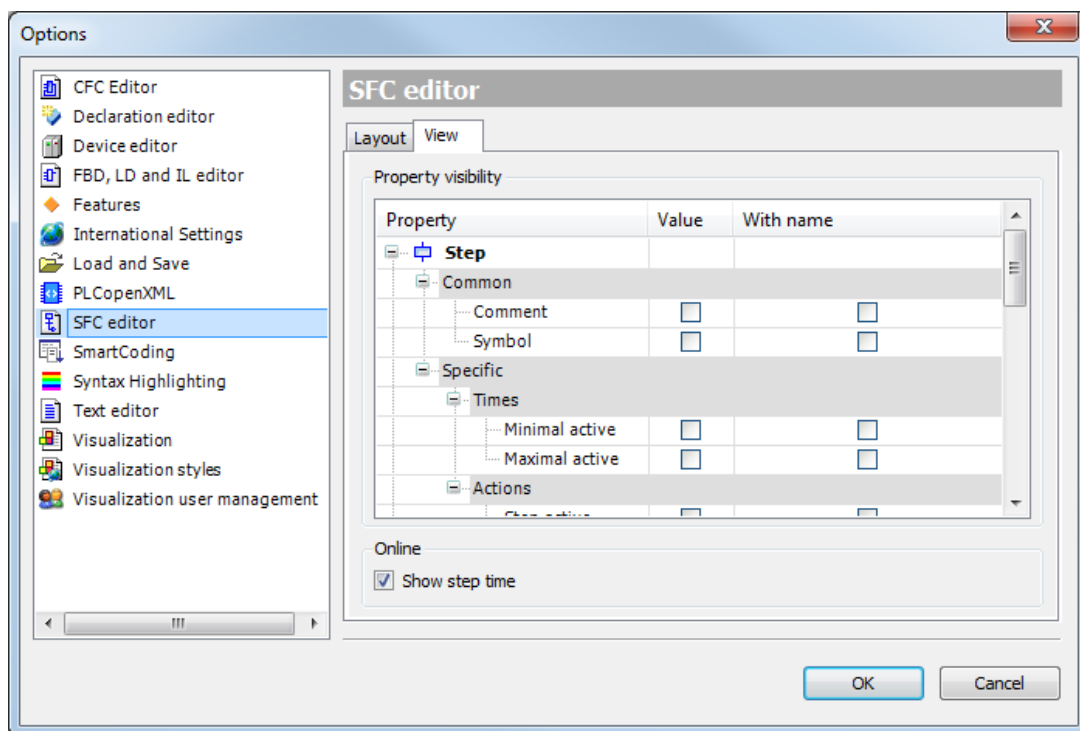


Figure 7-133. SFC Editor Options Dialog, Category View

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.

Property	Value
Common	
Name	s1
Comment	This is step s1.
Symbol	Write
Specific	
Initial step	<input type="checkbox"/>
Times	
Minimal active	t#2s
Maximal active	t#4s
Actions	
Step active	act1
Step entry	entry_at1
Step exit	exit_at1

SFC editor options.

Property	Value	With name
Step		
Common		
Comment	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Symbol	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Specific		
Times		
Minimal active	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Maximal active	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Actions		
Step active	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

SFC editor.

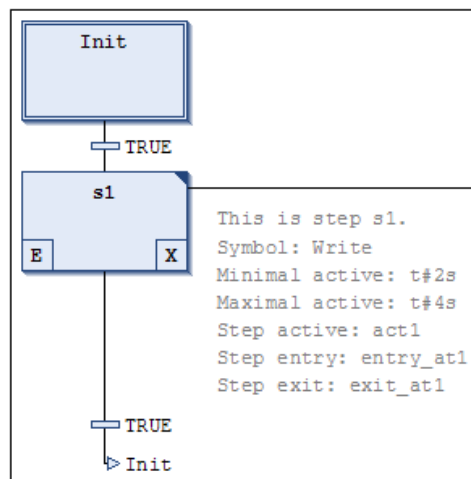


Figure 7-134. Property Visibility Example

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.

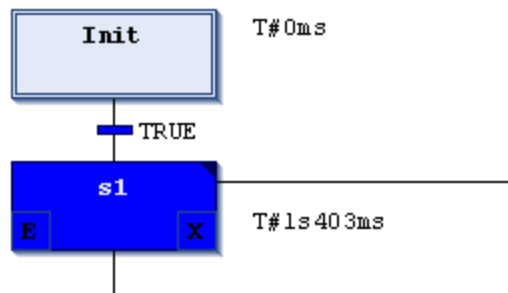


Figure 7-135. Step Time Displayed in Online Mode

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.

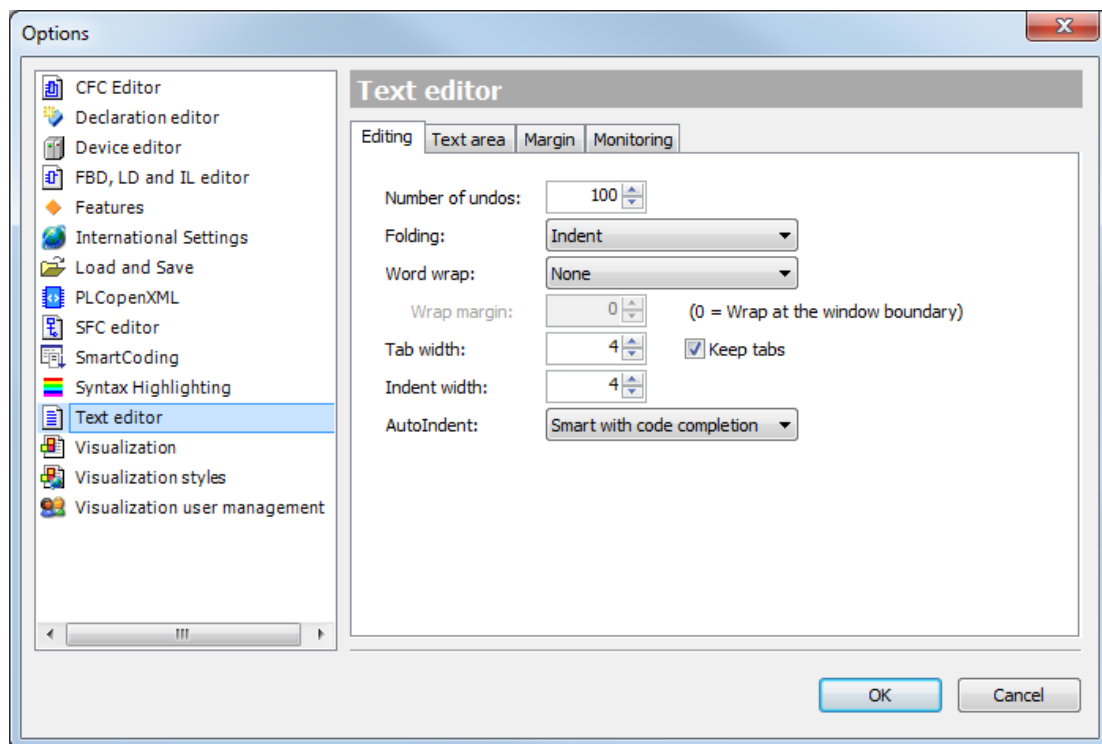


Figure 7-136. Options Dialog, Category Text Editor, Editing

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:

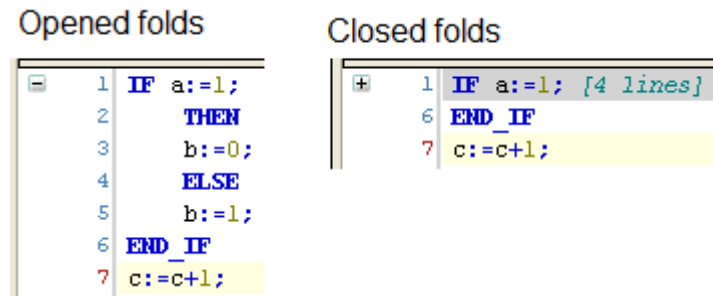


Figure 7-137. Opened and Closed Folds

- *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:

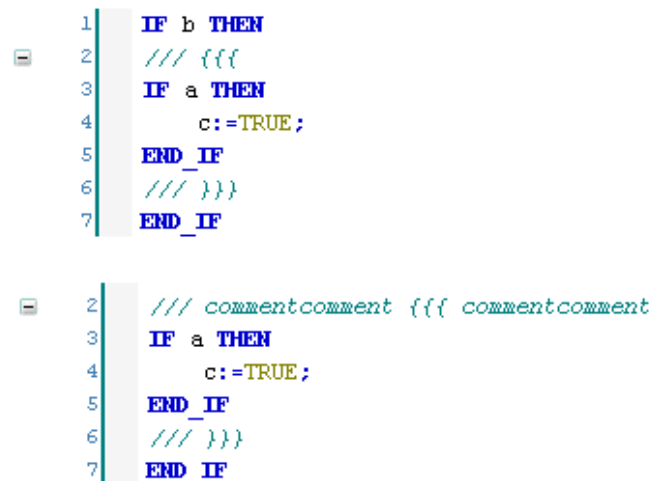


Figure 7-138. Explicit Folds Defined by Special Comments

- *Word Wrap*:
 - None: The line can be filled endlessly.
 - Soft: The lines will be broken at the boundary of the window.
 - Hard: The lines will be broken after the number of characters defined by *Wrap margin*. (A value of 0 means that the lines are wrapped at the window boundary).
- *Tab width*: Tab width in number of characters.
- *Indent width*: Indent width used in case of AutoIndenting that is the number of spaces at the beginning of a line.
- *Autoindent*: with the following options...
 - None: No automatic indenting. Editing always starts at the left margin of the editor's text area.

- *Smart*: Code lines following a keyword (for example VAR or IF) are automatically indented according to the Indent width defined above.
- *Smart with code completion*: When you enter a keyword like for example VAR or IF automatically the associated code lines following below are indented according to the Indent width defined above and additionally the terminating keyword is added (for example END_VAR or END_IF).
- *Keep Tabs*: If activated: A space which has been entered using the tab key according to the defined Tab width (see above) will not be resolved to single character spaces afterwards. So if the tab width gets changed later, the existing tab spaces will be corrected correspondingly.

Text Area

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

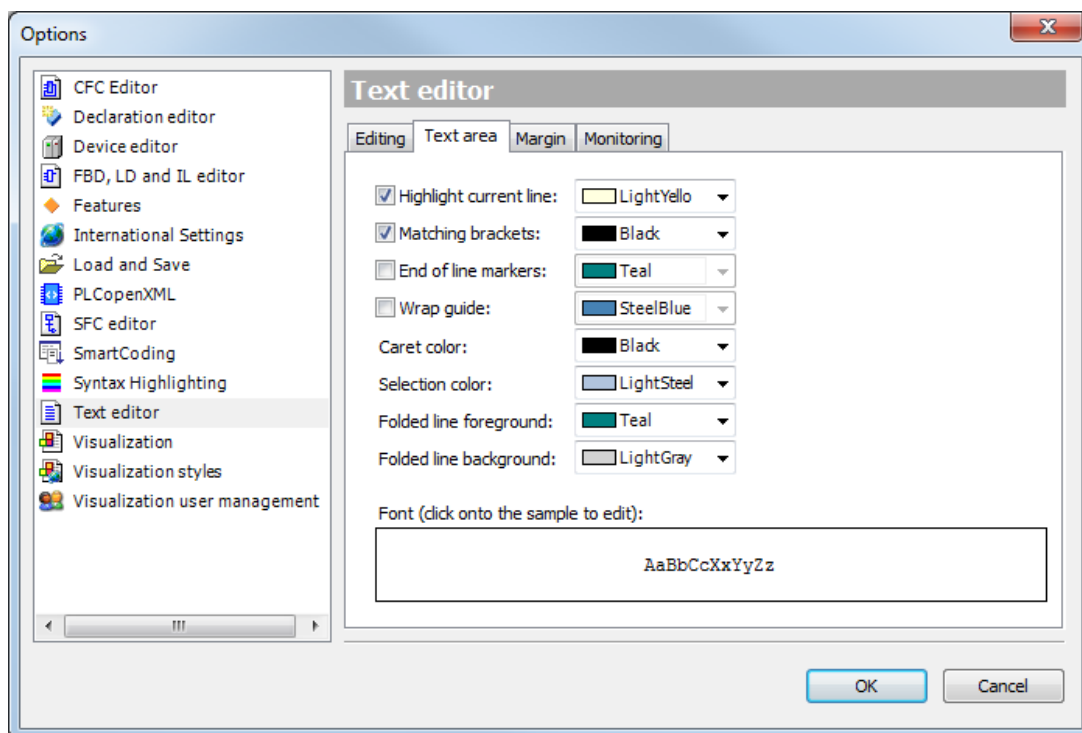


Figure 7-139. Options Dialog, Category Text Editor, Text Area

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 7-14. Visual Help to Edition

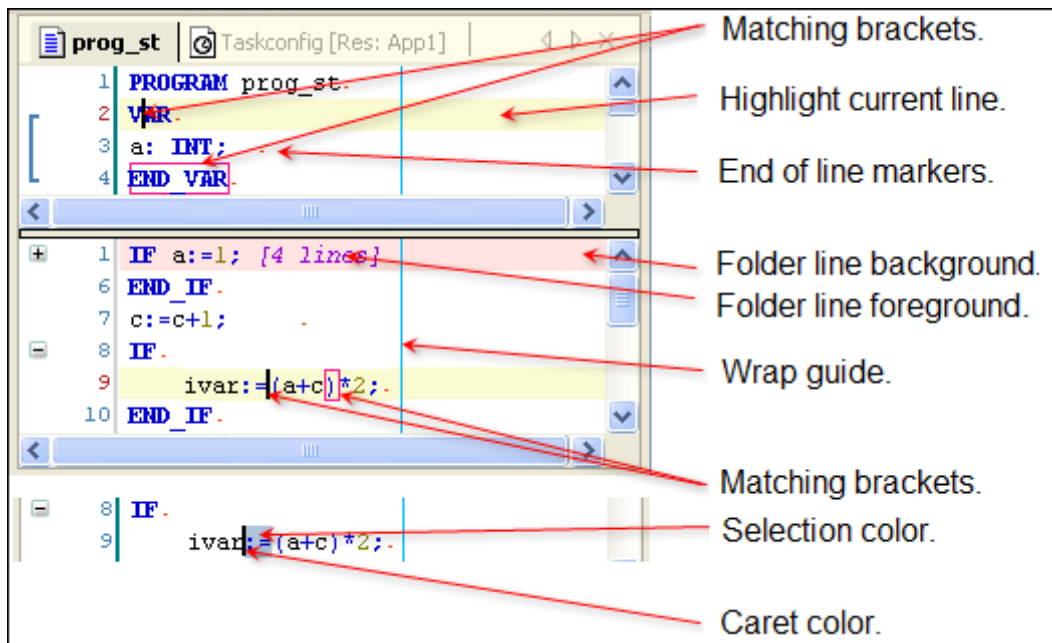


Figure 7-140. Examples for the Settings in the Text Area Dialog

Margin

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

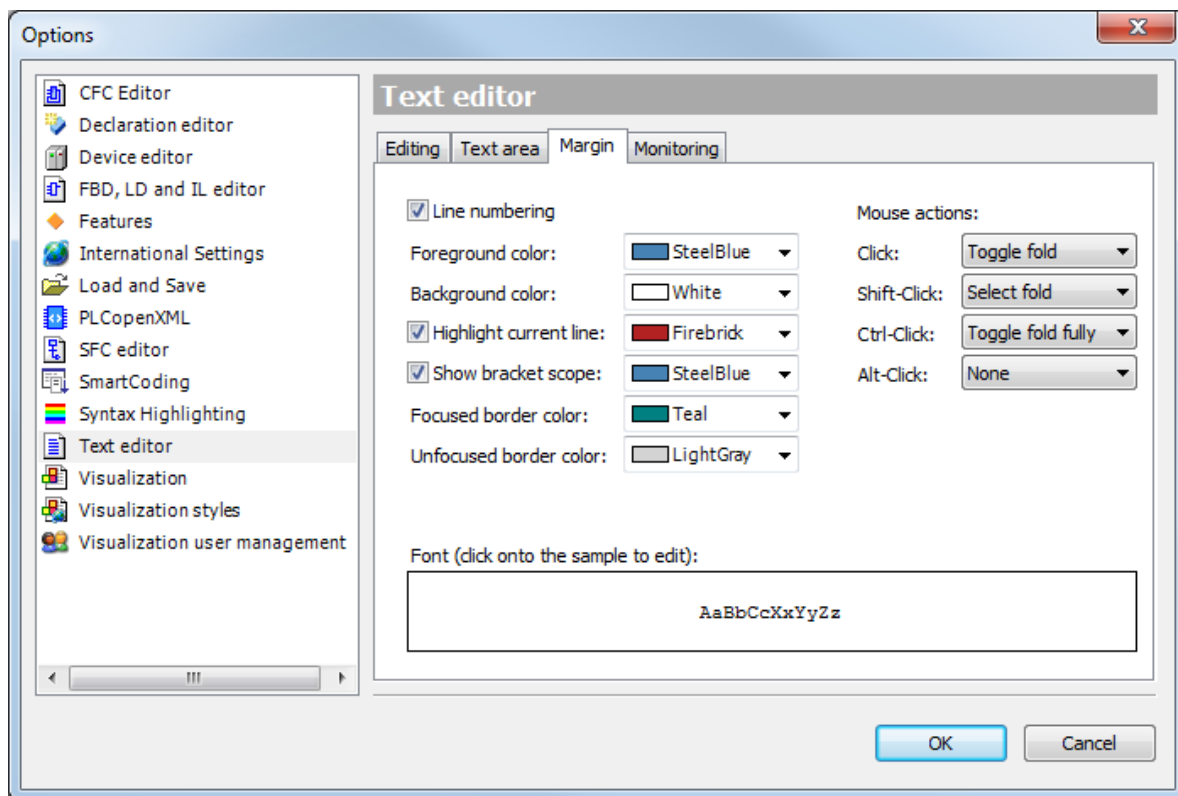


Figure 7-141. Options Dialog, Category Text Editor, Margin

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 7-15. Left Margin Options

Monitoring

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

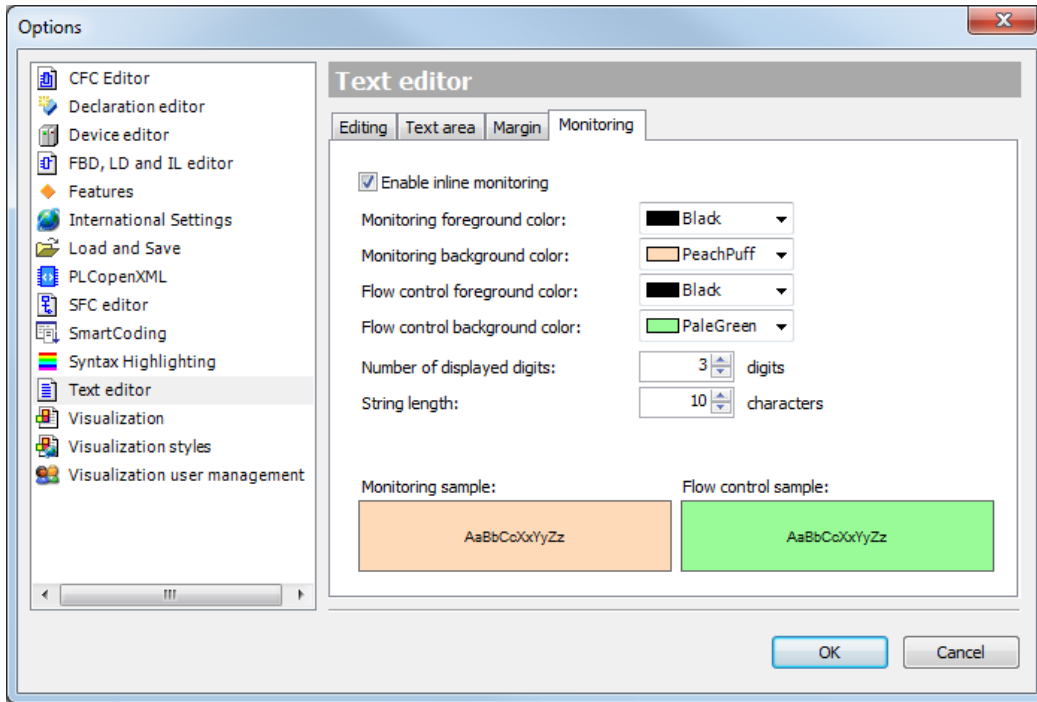


Figure 7-142. Options Dialog, Category Monitoring

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 7-16. Monitoring Fields

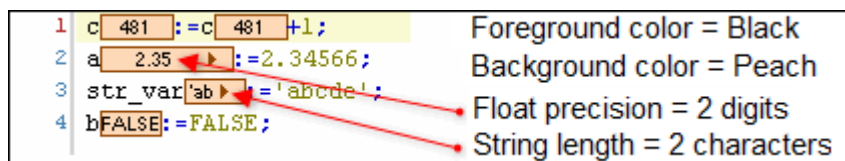


Figure 7-143. Example of Inline Monitoring in ST Editor

FBD, LD and IL Editor

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

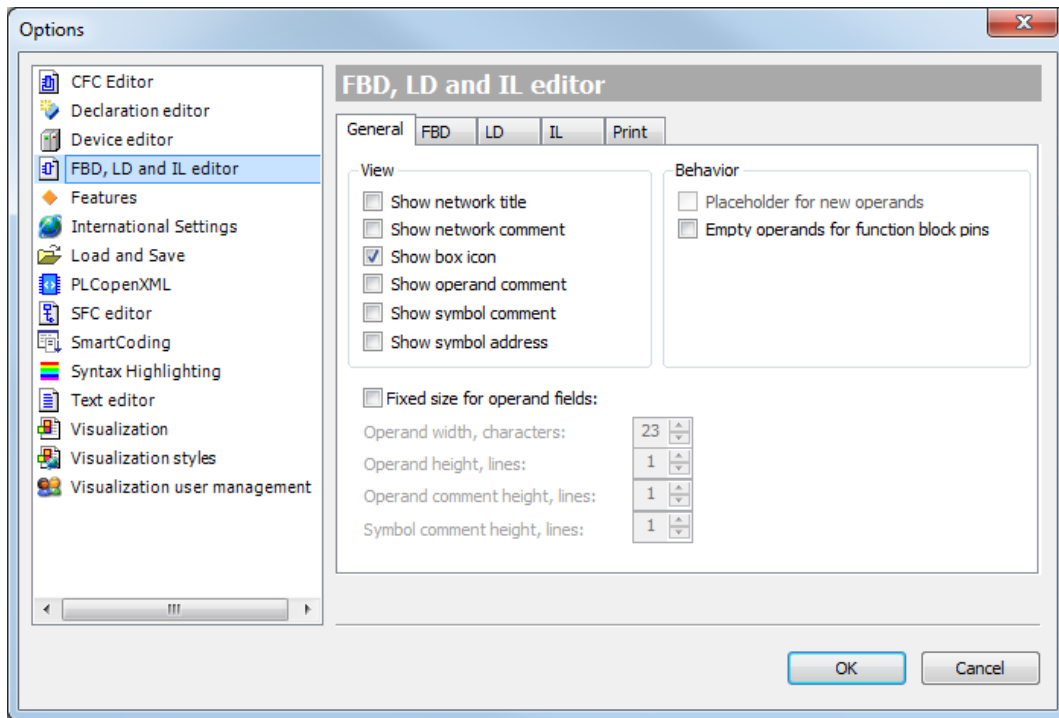


Figure 7-144. FBD, LD and IL Editors Viewing Options

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

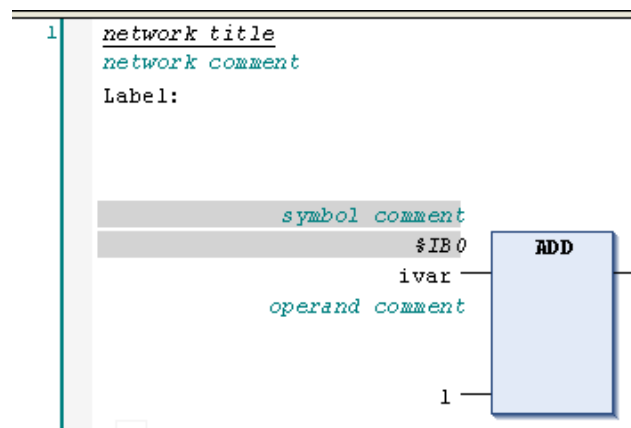


Figure 7-145. Components in the Editor's View

- *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.

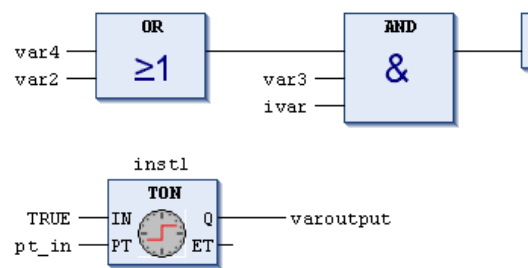


Figure 7-146. Boxes with icons for standard operators on a FBD network

- *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:

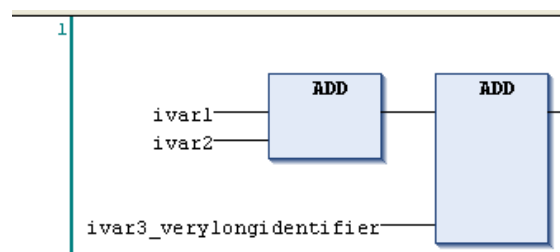


Figure 7-147. Enabled option

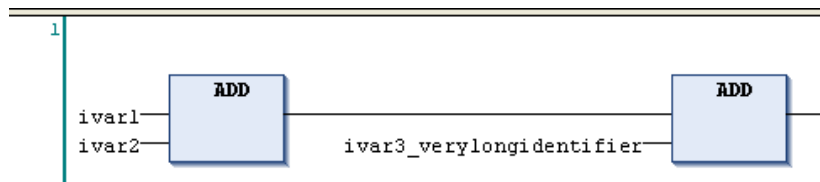


Figure 7-148. Disabled option

- *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.

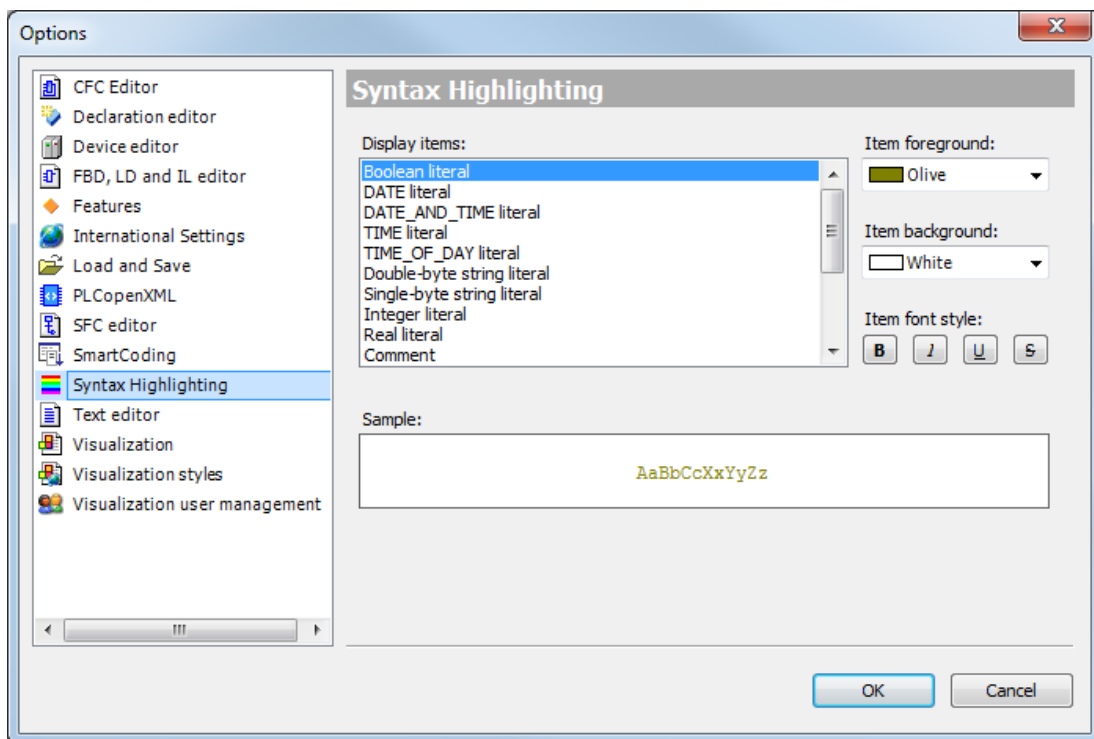


Figure 7-149. Options Dialog, Category Syntax Highlighting

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

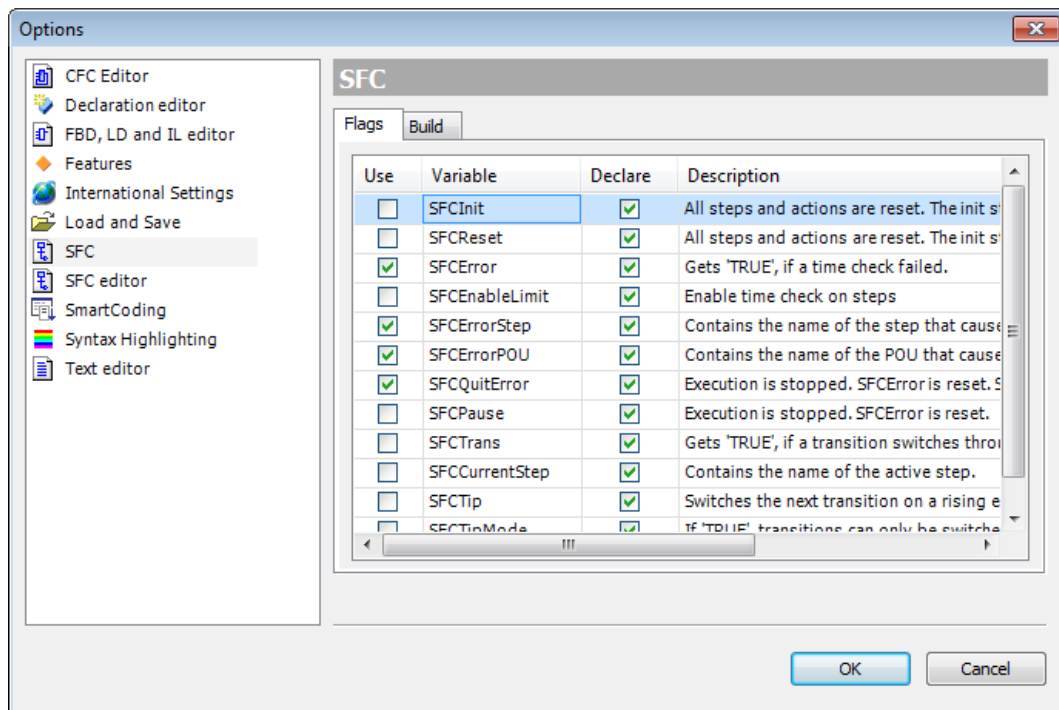


Figure 7-150. SFC Options Dialog, Flags

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.

The screenshot shows the SFC editor interface. At the top, a table lists error flags with their expressions, types, values, and prepared values. Below the table, a ladder logic diagram shows a transition labeled 'TRUE' leading to a step labeled 's1'. The step 's1' has an 'E' (Error) input and an 'X' (Exit) output. The step's code includes a timer and a write symbol.

Expression	Type	Value	Prepared value
t2111	BOOL	FALSE	
t222	BOOL	FALSE	
SFCErrror	BOOL	TRUE	
SFCErrrorPOU	STRING	'POU'	
SFCErrrorStep	STRING	's1'	
SFCQuitError	BOOL	FALSE	

The ladder logic diagram shows a transition labeled 'TRUE' leading to a step labeled 's1'. The step 's1' has an 'E' (Error) input and an 'X' (Exit) output. The step's code includes a timer and a write symbol:

```

T#2h57m54s995ms
This is step s1.
Symbol: Write
Minimal active: t#2s
Maximal active: t#4s
  
```

Figure 7-151. Example of Some SFC Error Flags in Online Mode of the Editor

Build

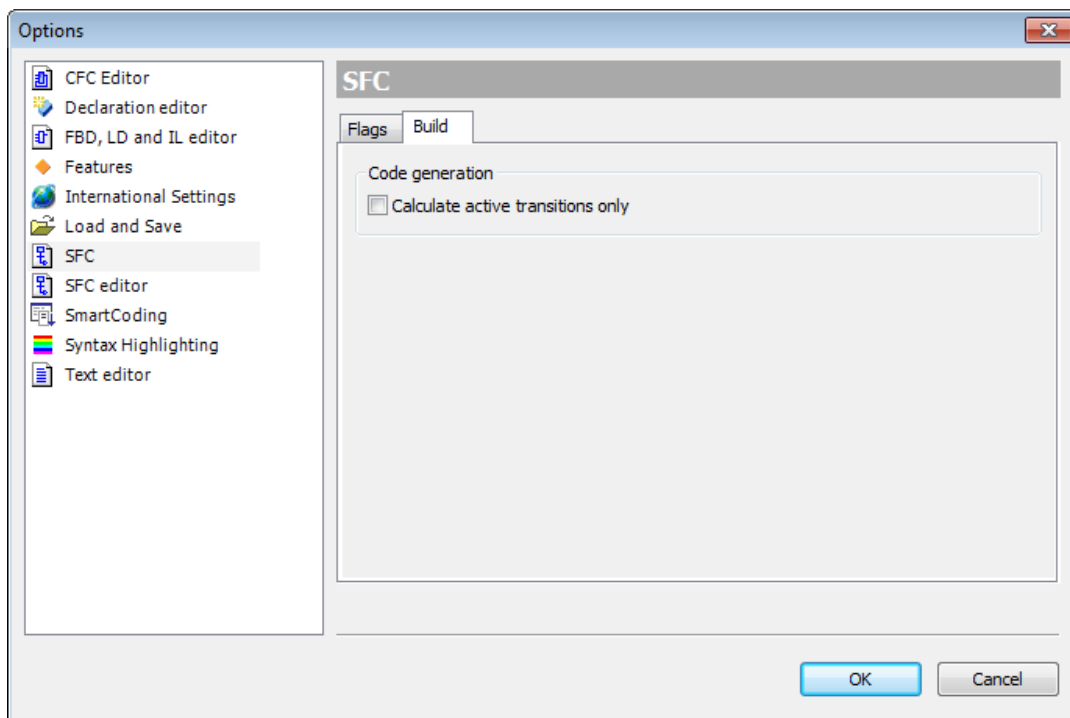


Figure 7-152. SFC Options Dialog, Build

- *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*.

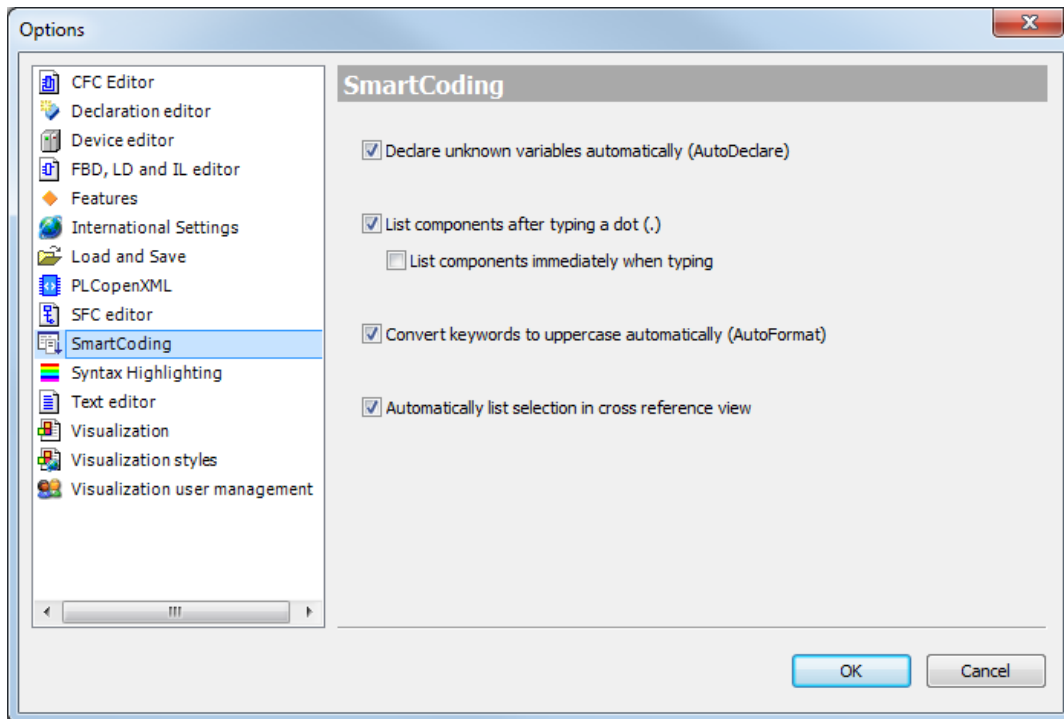


Figure 7-153. SmartCoding Options Dialog

- *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 "bVar:BOOL;".
- *Automatically list selection in cross reference view*: If this option is activated, the Cross Reference View always automatically lists the references of the variable currently selected in the active editor.

Visualization

This sub-dialog of the *Options* dialog consists of the tabs *General* and *Grid* allowing common settings for working on visualizations in the programming system.

General

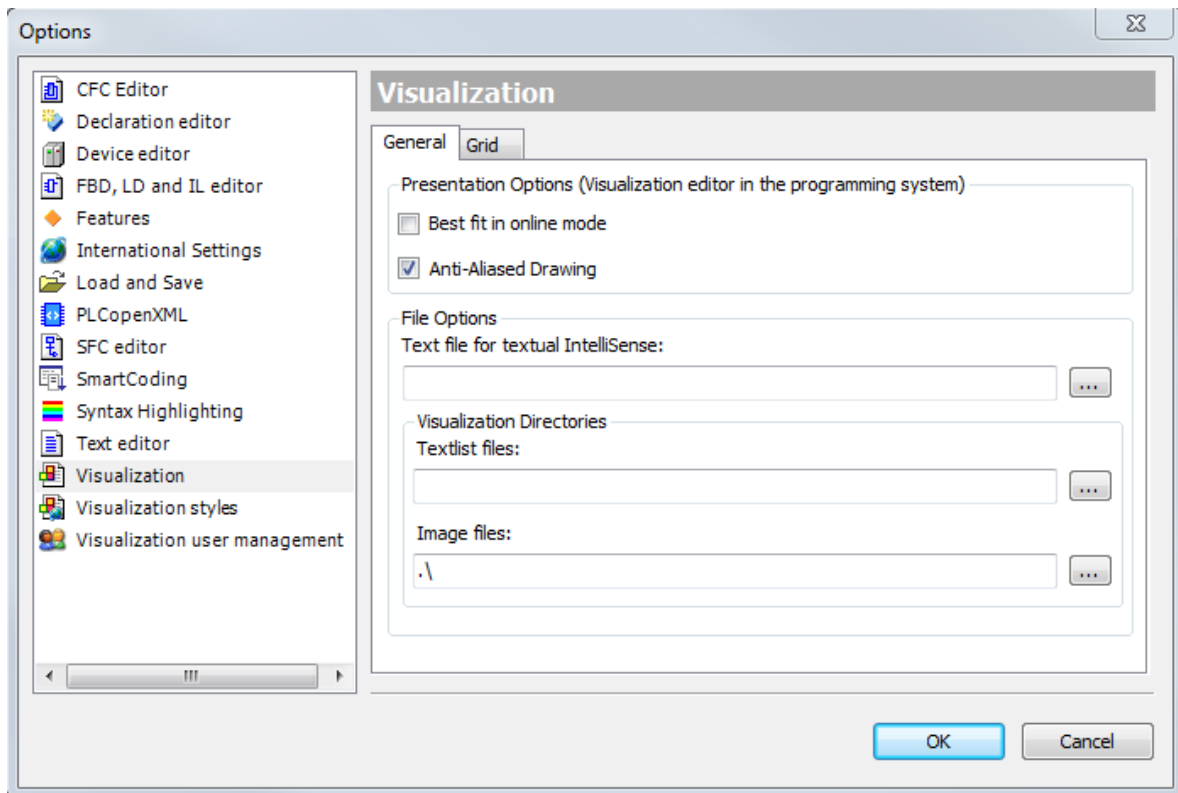


Figure 7-154. Visualization Options Dialog, General

Presentation Options (Visualization editor in the programming system):

- *Best fit in online mode:* Activate this checkbox for the visualization being scaled optimally according to the actual size of the online visualization window within the programming system.
- *Antialiased Drawing:* Activate this checkbox, if antialiasing should be used when drawing the visualizations within the visualization editor view of the programming system (offline or online).

File Options:

Text file for textual IntelliSense: In the related edit field you may enter the name (including the path) of the file (of type .csv) providing the proposals that will be made by the List Components function, when you enter a string in the edit field of the item Texts\Text within the properties of a visual element. This text file may stem for example from the export of a global textlist. The text file has to be designed as table, its structure must follow the structure of textlists.

Visualization Directories:

Here you can define default directory paths, which however still might be modified project-specifically in the Project Settings, category Visualization. Finally valid for the project are the **Project Settings**



Textlist files: Use button  to open the standard dialog for specifying an existing or new directory for language files (see chapter Textlists), which should be available for text and language configuration within the visualization. The files found in this directory will be available for selection when a textlist should be imported, e.g. after having been edited by an external translator. Only one directory can be specified here.

Image files: Use button  to open the standard dialog for specifying one or several existing or new directory for image files (image pools), which should be available for the use in the visualizations. If entering several directories, separate by semicolons.

Grid

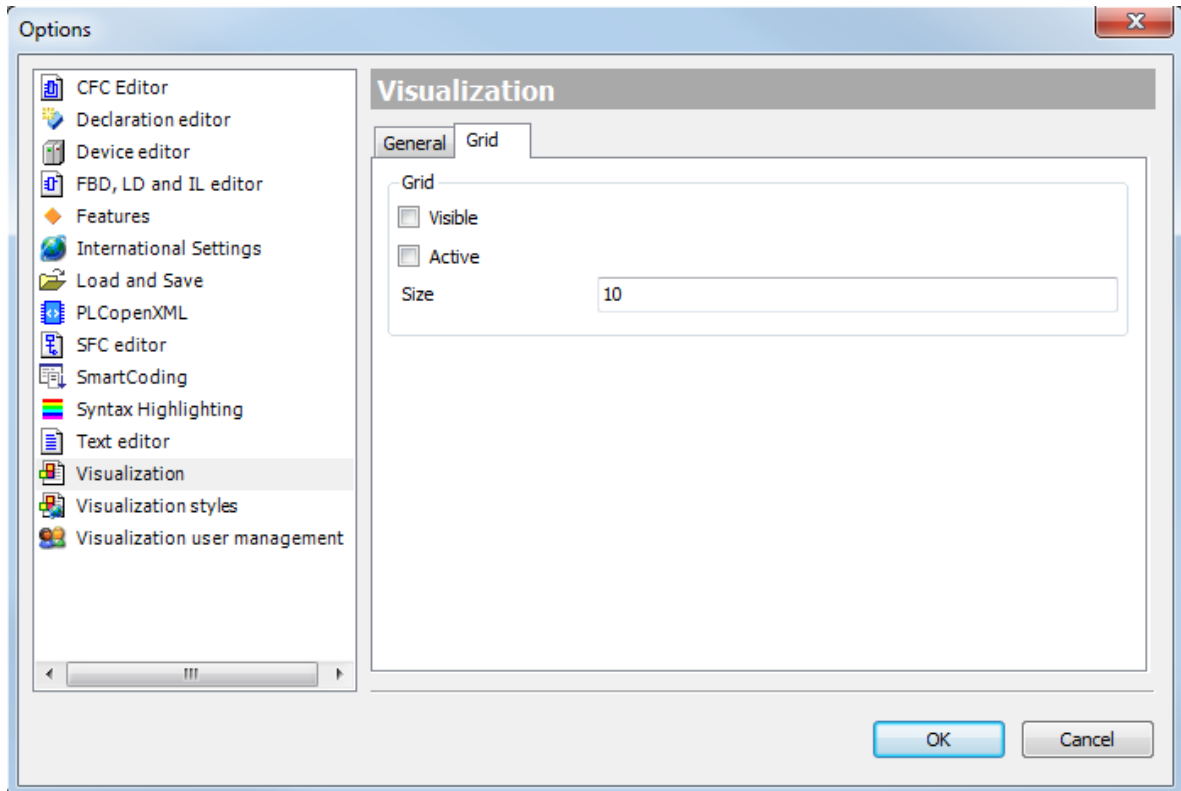


Figure 7-155. Visualization Options Dialog, Grid

Visible: If this option is activated, a grid according to the Size defined below will be displayed in the visualization editor window in order to ease optic orientation.

Active: If this option is activated, visualization elements can only be positioned according to the grid points defined below in Size, no matter whether the grid is visible or just active. When inserting or moving an element this means that the center of the element can only be placed on a grid point. When modifying an element, it means that the small rectangle at an edge of the element, which gets shifted in order to change the size or form of the element, can only be placed on a grid point.

Size: Vertical and horizontal distance between the grid points in pixels.

Visualization do Gerenciamento do Usuário

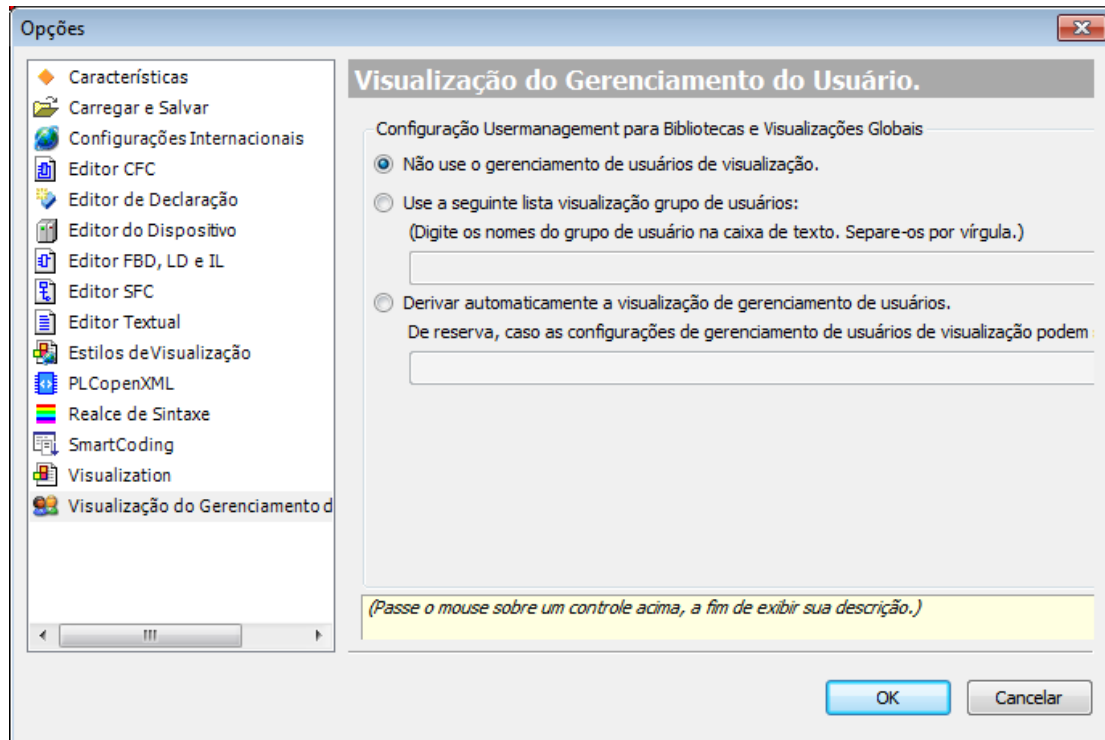


Figure 7-156. Visualization User Management

User management configuration for libraries and global visualizations: the option is used for visualizations within libraries and for global visualizations located under the POU's tree.

- *Use no visualization user management* : tick this option then the affected visualizations will behave as if no user management configuration is assigned.
- *Use the following visualization user group list* : tick this option then the affected visualizations will use the groups listed in the text field. It can be edited manually or by using the button **Export groups** for global visualization in dialog **User Management**.
- *Derive visualization user management automatically:* tick this option then the affected visualizations will derive the user management configuration from the selected visualization manager. The selection list provides all actually installed Visualization Managers of the project. If this is not possible then the user group list from option *Use the following visualization user group list* will be used.

The user management configuration of a visualizations under the device tree have to be done in **User Management** of the associated visualization manager.

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.

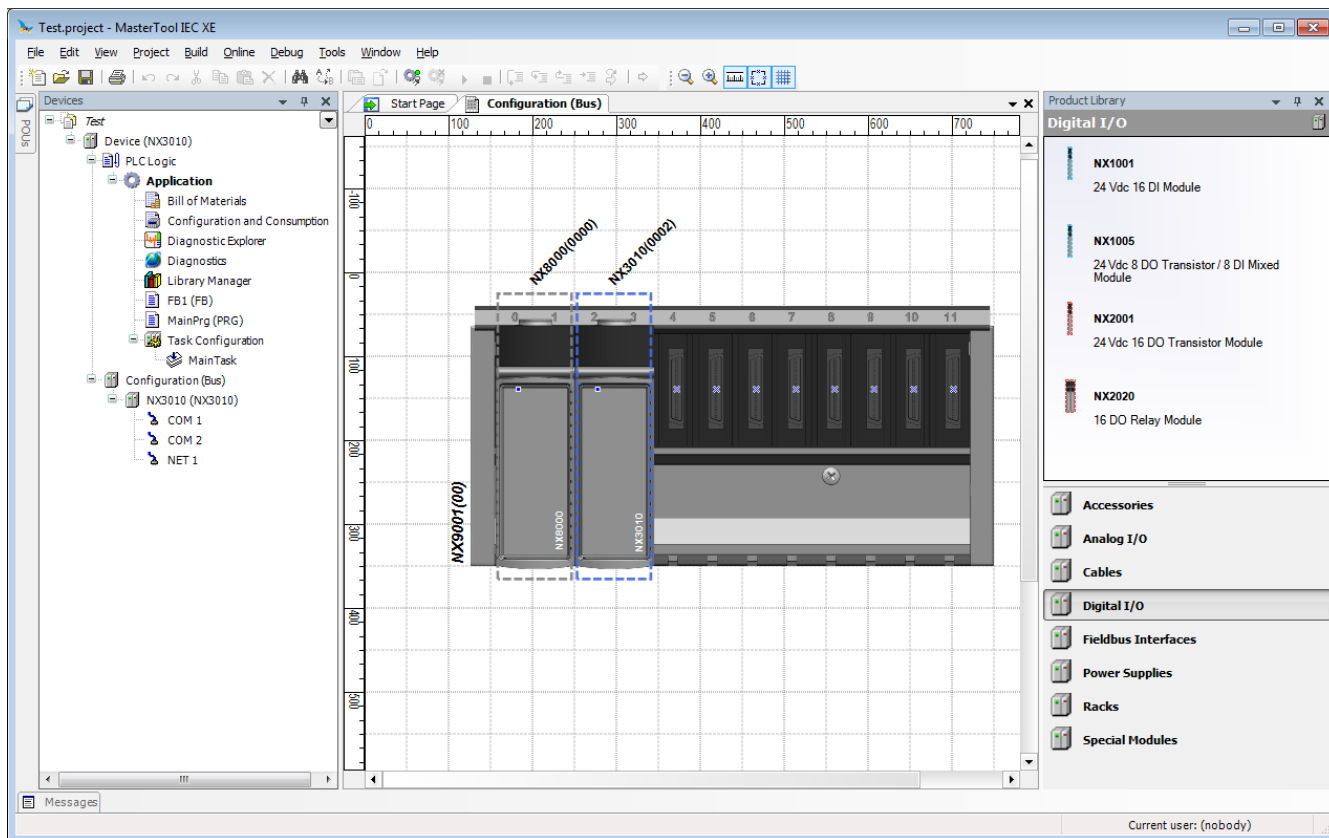


Figure 8-1. Bus Editor

Add Device

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

If the added device is not properly connected to the bus, it will be marked with a rectangle and a diagonal red bar, as seen in Figure 8-2.

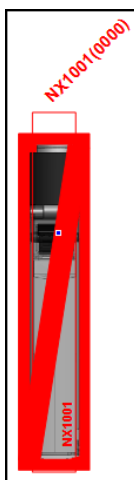


Figure 8-2. Disconnected Device

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 8-3:

Select the Outputs		Select the Inputs	
Name	Type	Name	Type
<input type="checkbox"/> HSC Input 00 Command		<input checked="" type="checkbox"/> Digital Inputs	
High Speed Counter Input 00 Command	BYTE	Digital Inputs - Byte 0	BYTE
<input type="checkbox"/> HSC Input 00 Preset Value		Digital Inputs - Byte 1	BYTE
High Speed Counter Input 00 Preset Value	DINT	<input type="checkbox"/> HSC Input 00 Status	
<input type="checkbox"/> HSC Input 01 Command		High Speed Counter Input 00 Status	BYTE
High Speed Counter Input 01 Command	BYTE	<input type="checkbox"/> HSC Input 00 Current Value	
<input type="checkbox"/> HSC Input 01 Preset Value		High Speed Counter Input 00 Current	DINT
High Speed Counter Input 01 Preset Value	DINT	<input type="checkbox"/> HSC Input 01 Status	
<input type="checkbox"/> Pulse-Catch Reset		High Speed Counter Input 01 Status	BYTE
Pulse-Catch Reset - Byte 0	BYTE	<input type="checkbox"/> HSC Input 01 Current Value	
Pulse-Catch Reset - Byte 1	BYTE	High Speed Counter Input 01 Current	DINT
		<input type="checkbox"/> Input 02 Period	
		Input 02 Period	DWORD

Figure 8-3. 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 8-4:

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 8-4. Module Parameters

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

Bus: I/O Mapping

The Figure 8-5, shows the map of all the module inputs/outputs.

Process Data | Module Parameters | **Bus I/O Mapping**

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
		Digital Inputs - Byte 0	%IB0	BYTE		Input state.
		Input 00	%IX0.0	BOOL		
		Input 01	%IX0.1	BOOL		
		Input 02	%IX0.2	BOOL		
		Input 03	%IX0.3	BOOL		
		Input 04	%IX0.4	BOOL		
		Input 05	%IX0.5	BOOL		
		Input 06	%IX0.6	BOOL		
		Input 07	%IX0.7	BOOL		
		Digital Inputs - Byte 1	%IB1	BYTE		Input state.
		Input 10	%IX1.0	BOOL		
		Input 11	%IX1.1	BOOL		
		Input 12	%IX1.2	BOOL		
		Input 13	%IX1.3	BOOL		
		Input 14	%IX1.4	BOOL		
		Input 15	%IX1.5	BOOL		
		Input 16	%IX1.6	BOOL		
		Input 17	%IX1.7	BOOL		

Reset mapping Always update variables

IEC Objects

Variable	Mapping	Type
NX1001		NextoSlave

= Create new variable
 = Map to existing variable

Figure 8-5. 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 8-6. This list can also be used for purchase orders.

The following modules and products are shown in this list:

- CPUs
- Input and Output Modules
- Fieldbus Interface Modules
- Power Source Modules
- Special Modules (bus expansion and redundancy)
- PROFIBUS Slaves
- PROFIBUS Input and Output Modules
- PROFIBUS Modules base (consult PROFIBUS Modules Base – Ponto Series)
- Backplanes

- Accessories
- Cables

Device	Product Description	Count
NX1001	24 Vdc 16 DI Module	1
NX1005	24 Vdc 8 DO Transistor / 8 DI Mixed Module	1
NX2001	24 Vdc 16 DO Transistor Module	1
NX2020	16 DO Relay Module	1
NX3030	High-speed CPU, 2 Ethernet Ports, 2 Serial Channels, Memory Card Interface, Remote Rack Expansion and Redundancy Support	1
NX5000	Ethernet Module	1
NX5001	PROFIBUS DP Master	1
NX6000	8 Ch. Analog Input Module	1
NX6100	4 Ch. Analog Output Module	1
NX8000	30 W 24 Vdc Power Supply Module	1
NX9002	16-Slot Backplane Rack	1
NX9102	Backplane Connector Cover	1

Show Backplane Connector Cover

Figure 8-6. 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.

PROFIBUS Modules Base – Ponto Series

PROFIBUS modules of the Ponto series require bases. Bases are modular elements that make up the bus. They are assembled in TS35 rails and distribute power, bus signals and I/O signals. Bases have terminal blocks in spring or screw options for field wiring and optional protection fuses. Base selection depends on the module being used. The user must consult the module's datasheet to determine the available and suitable bases.

To select the base to be used by a given module, open its configuration screen in the *General Parameters* tab, then choose the base in the field *Module Base*, as in Figure 8-7.

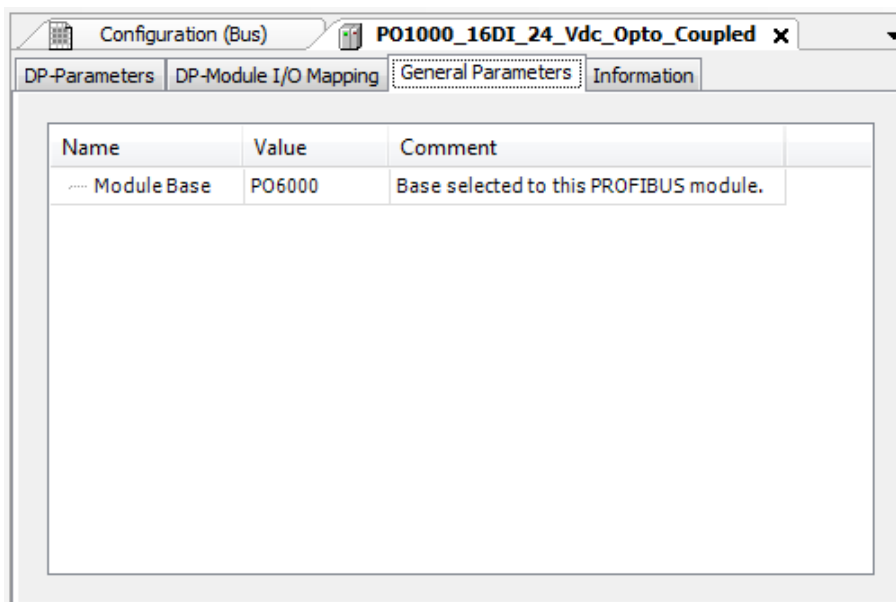


Figure 8-7. Module Base Configuration

Configuration and Consumption

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

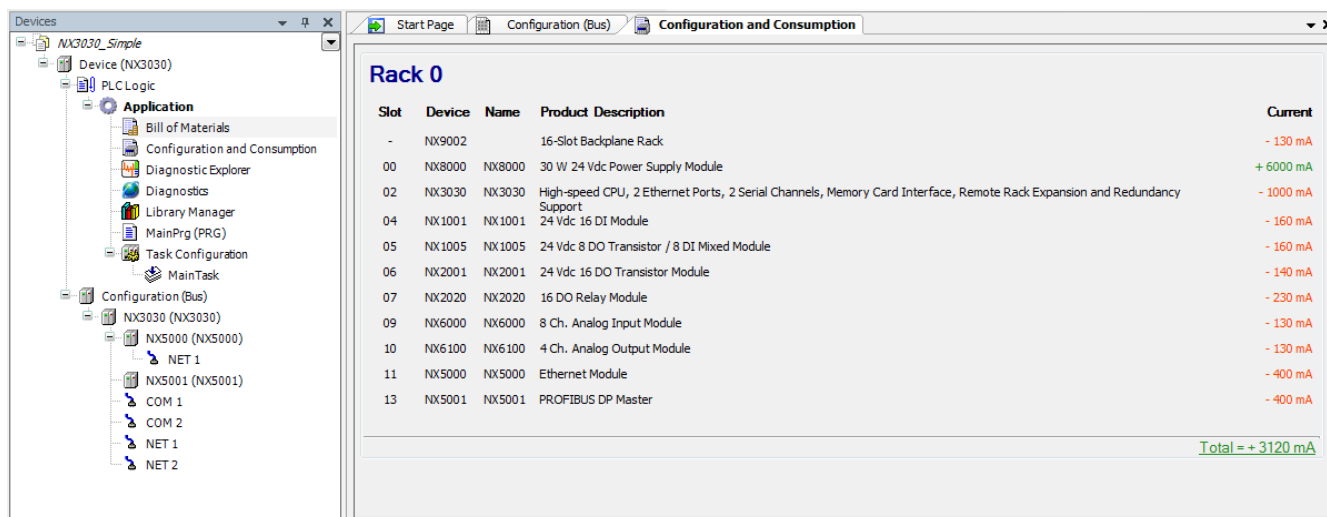


Figure 8-8. Configuration and Consumption

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

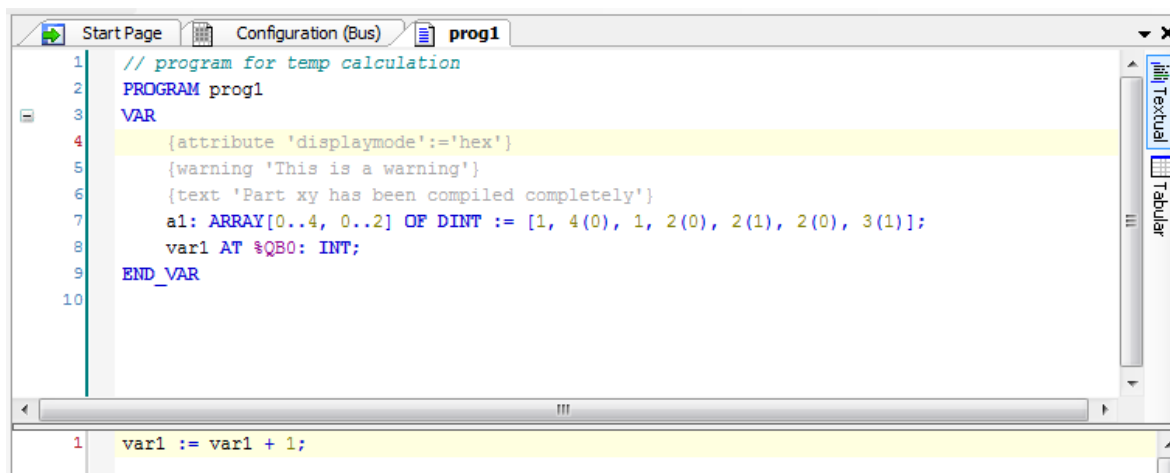


Figure 8-9. Textual Editor View

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

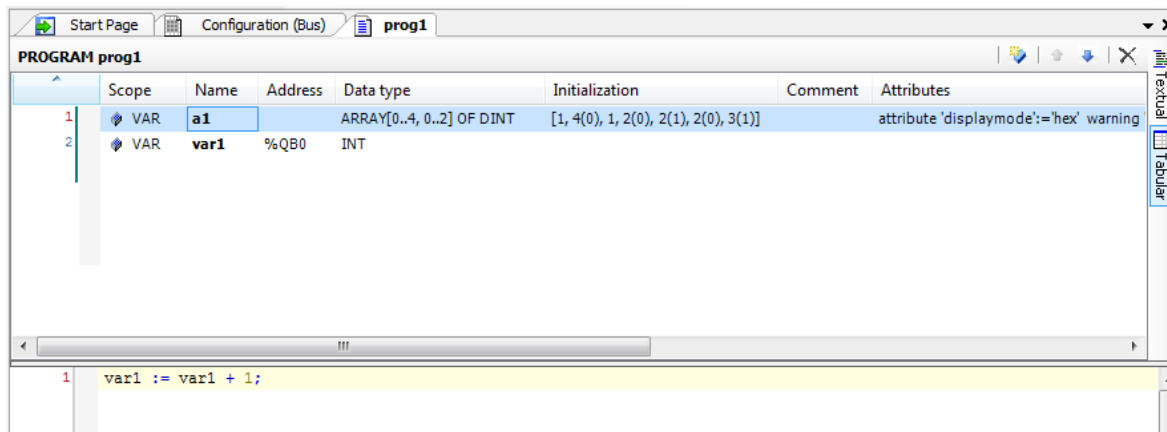


Figure 8-10. Tabular Editor View

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.

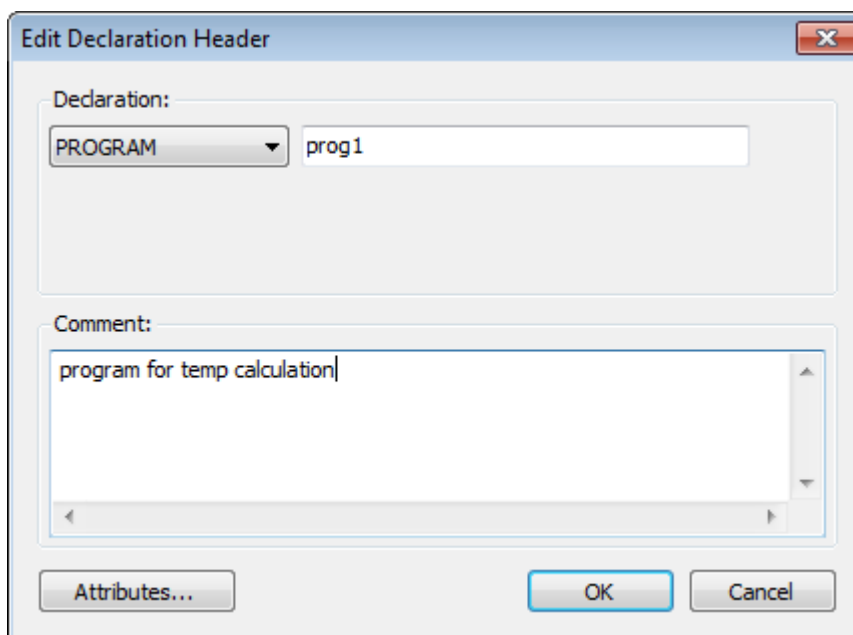



Figure 8-11. Edit Declaration Header Dialog

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.

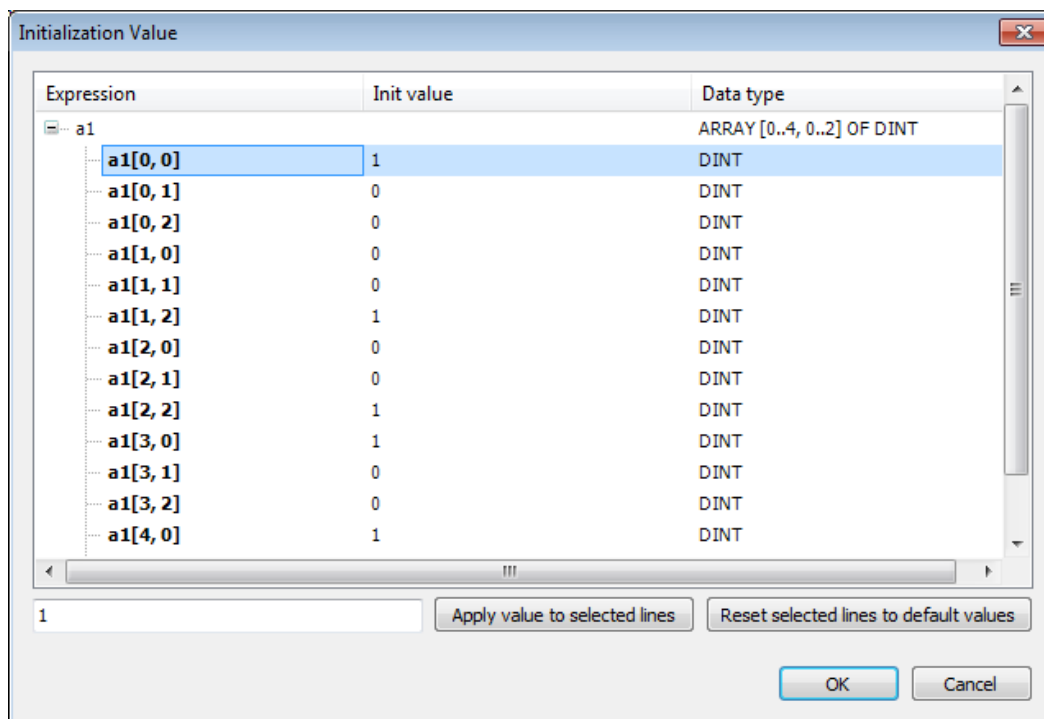


Figure 8-12. Initialization Value Dialog

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.

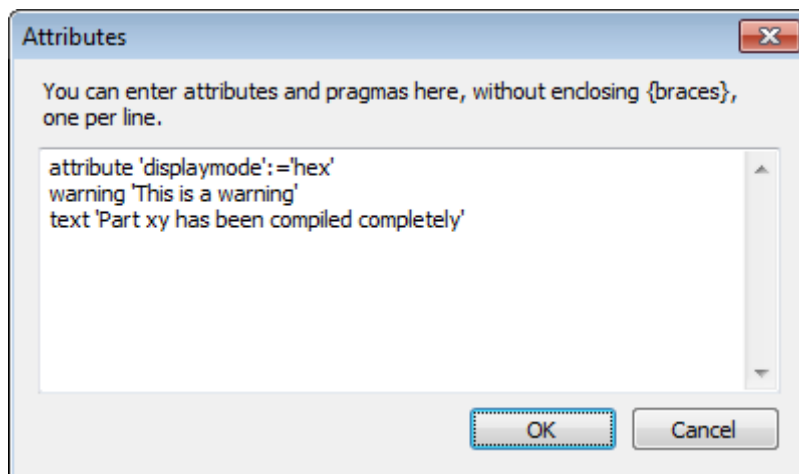


Figure 8-13. Attributes

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 **** 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 8-14). 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 8-14.

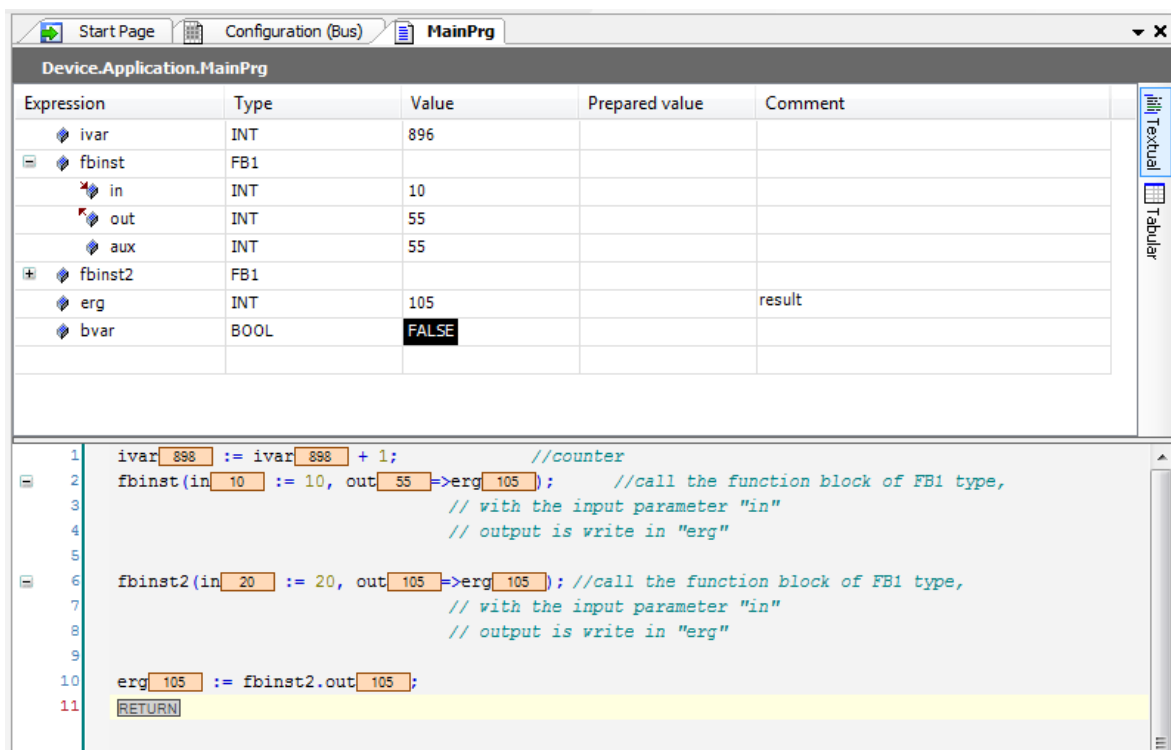


Figure 8-14. Declaration Editor in the Upper Part of a program object MainPrg, Online View

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.

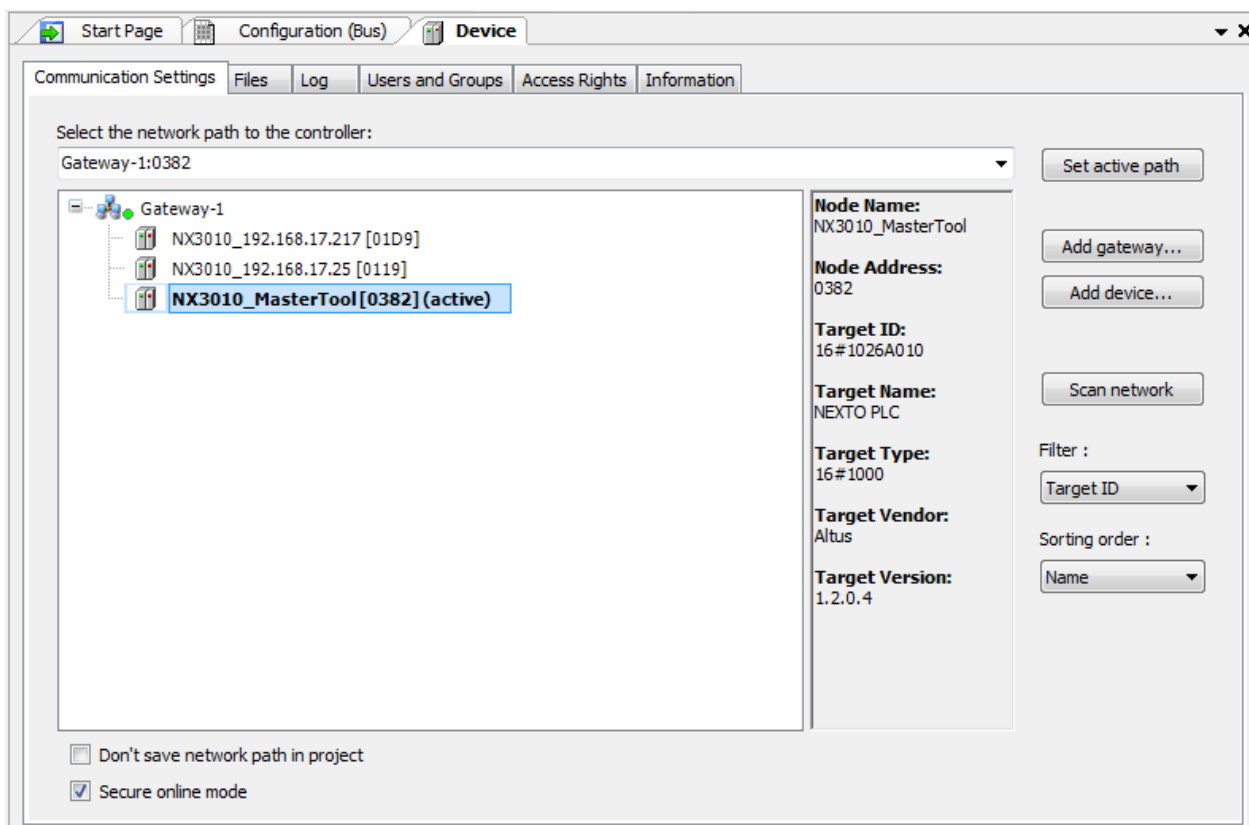






Figure 8-15. Device Dialog, Communication Settings

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.

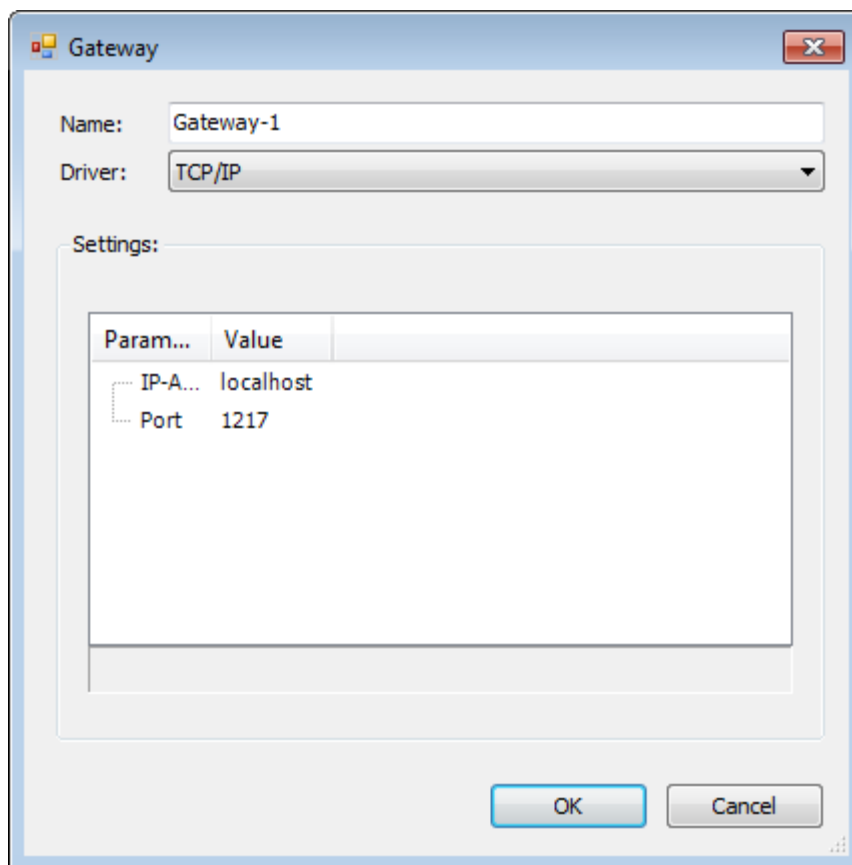


Figure 8-16. Gateway Dialog

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.

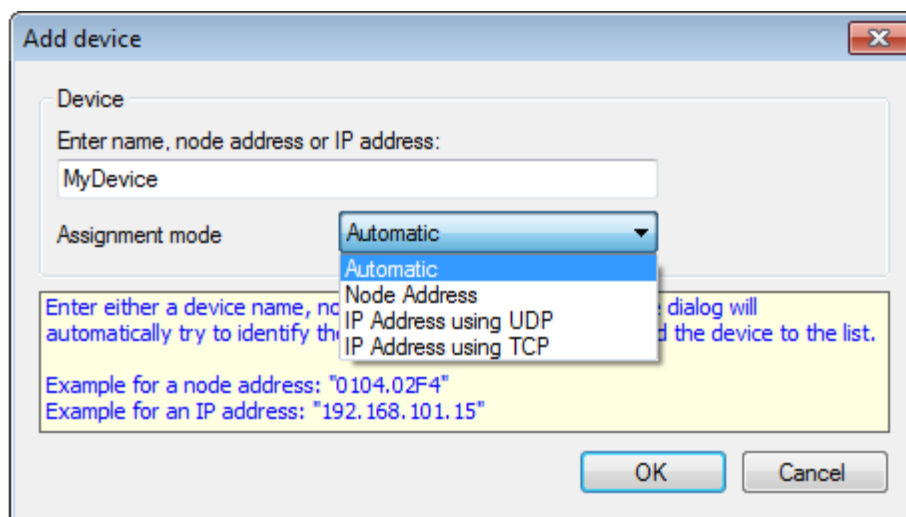


Figure 8-17. Add Device Dialog

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 (e.g. 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 (e.g. 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 8-18 where Gateway (GW0) of the computer (PC10) is being used to access the device from same subnet (PLC20). In Figure 8-18 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.

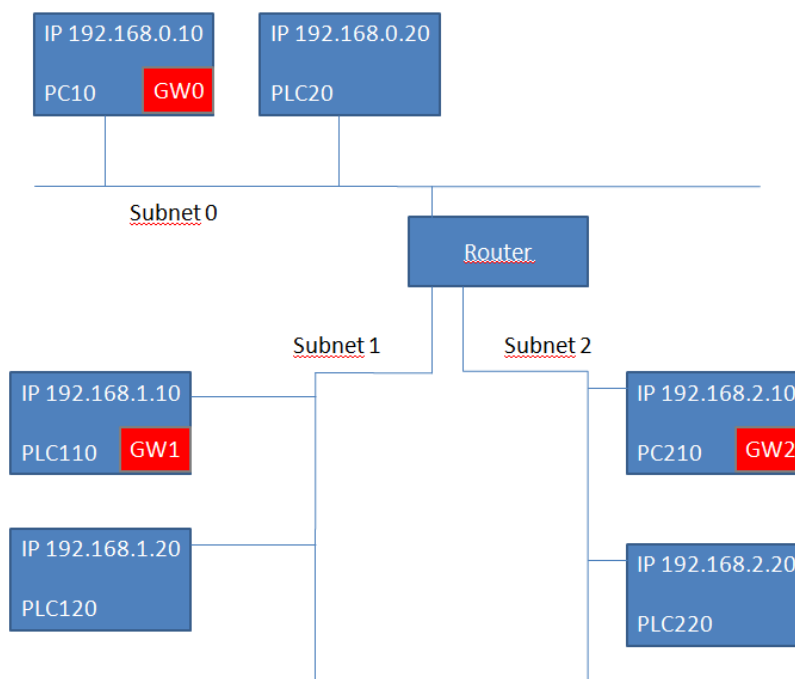


Figure 8-18. Remote Gateway

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 8-18, there is no gateway running on the subnet 1. This way to access the PLC110 it is possible to configure the gateway of the device itself (GW1) which will be responsible for handling the packages with the computer on the subnet 0 for example.

In both cases, it is necessary to know what the device IP address is and put it into the parameter *IP-Address* when it added a new gateway. By using one of these two paths it is possible that the PC10 access PLC120 and PLC220 in different subnets than yours. For details on adding gateway see **Communication Settings**.

NOTE: In some cases, when different subnet than the PLC, you must perform a connectivity test using the command "ping", through the Windows prompt, between the computer and the PLC to this one appears in the *Communication settings* window.

ATTENTION:

All Nexto series CPUs can be used as a remote gateway in different subnets of the computer where the MasterTool IEC XE is running. However, this operation causes performance loss of your Ethernet interfaces due to treatment and redirection of packages sent to other devices on the same network. Whenever a device is accessed on a network other than the computer where there is a gateway to redirect packets transmitted by MasterTool IEC XE, it is always recommended to use the IP address of the destination avoiding this unnecessary traffic to other equipment not involved in communication.

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.

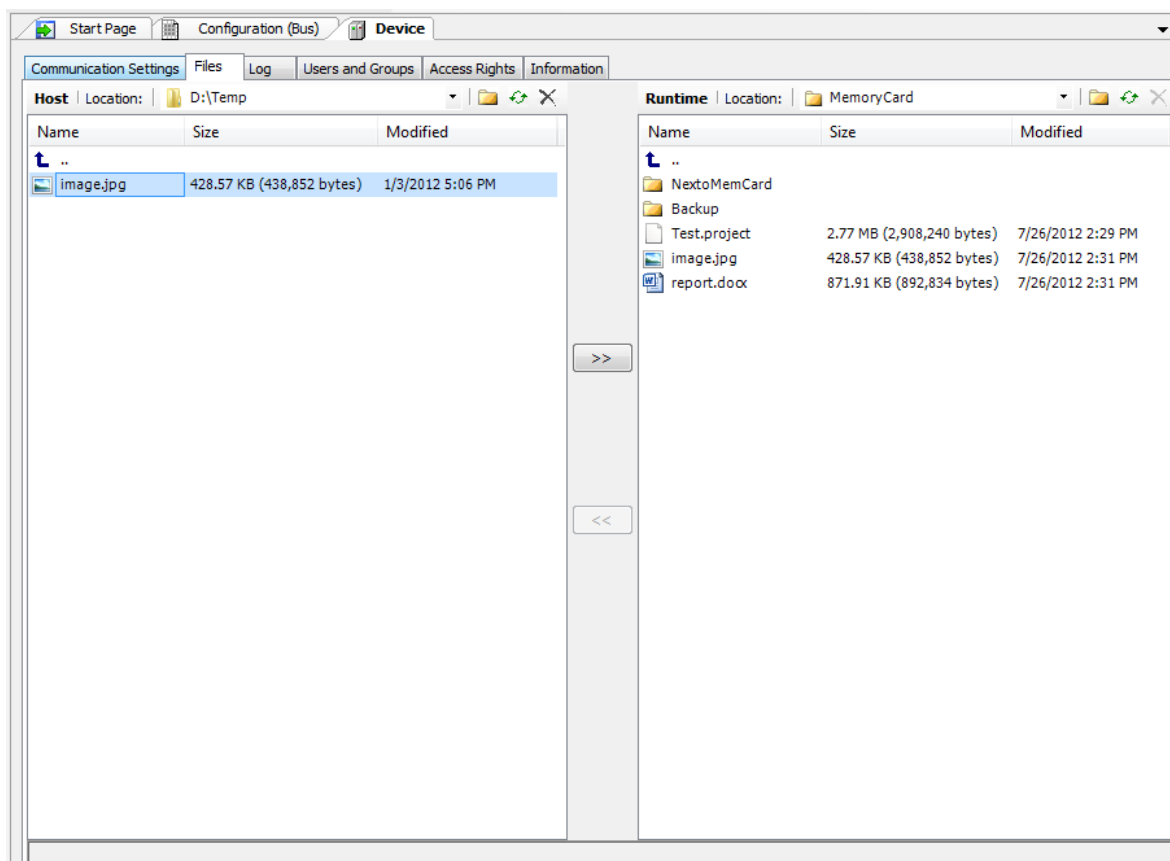




Figure 8-19. Device dialog, Files

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.

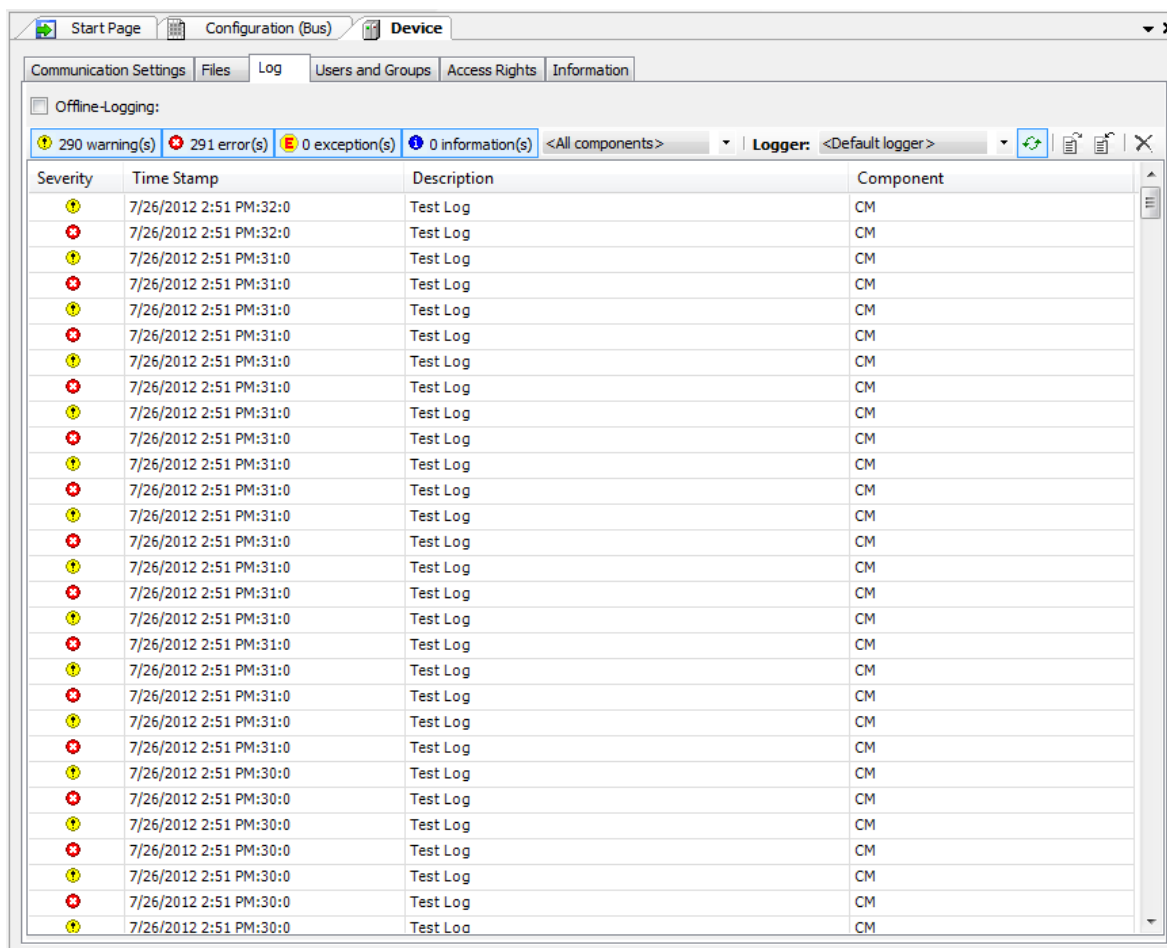




Figure 8-20. Device Dialog, Log



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 log entries. The default setting is “<Default Logger>”, which is defined by the runtime system and currently is identical with "PlcLog" for runtime system in Nexto Series CPUs. Currently the user can manually update the list by using  button. By clicking once, it will be pressed () and will update the log automatically. By clicking again, the display is disabled (new log entries will only be shown when clicking again).

NOTE: While the update button is pressed, the programmer and the CPU will be connected, even if the Logout command is executed. In this case, to provide the communication and / or sending project to other CPUs, the refresh button should be disabled.

The list can be exported to or imported from a XML-file. To export use  button to get the standard dialog for saving a file. The file filter is set to "xml-files (*.xml)". The log-file will be stored with the specified file name with extension ".xml" in the chosen directory. To view log entries stored in an xml-file which might have been exported in this way, use  button. The standard dialog for browsing for a file will open. In addition, here the filter is set to "xml-files (*.xml)". Choose the desired log-file and its entries will be displayed in a separate window.

For clearing the current log table, that is removing all displayed entries, use  button.

If option *Offline-Logging* is activated, also certain offline actions will be logged, which do not refer to the connection with the PLC.

OPC Communication Editors

Nexto Series CPUs support the OPC DA (Open Platform Communications Data Access) communication technology. This open communication platform was developed to be the industrial communication standard. Based on a client/server architecture, it offers several advantages in project development and ease of communication with automation systems. Below are presented the configuration environments of the OPC communication in MasterTool IEC XE software. For further information on OPC Communication, consult Nexto Series CPUs User Manual – MU214605.

OPC Configuration

The PLC configuration is made within MasterTool IEC XE through an option available at *Online* menu. It is necessary that MasterTool IEC XE be executed as administrator. Figure 8-21 presents OPC Communication's configuration interface. The fields are described in Table 8-1.

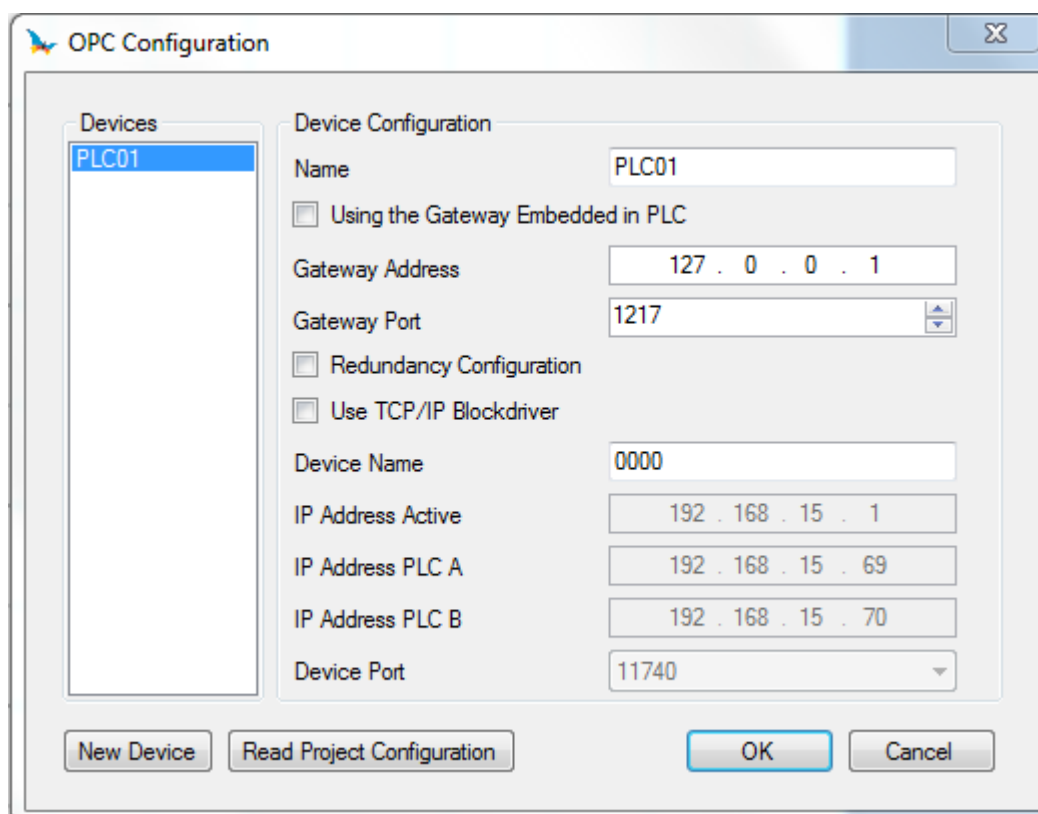


Figure 8-21. OPC Server Configuration

Device Configuration	Description	Factory Default	Possibilities
Name	PLC description within OPC Server configuration file. This field may have any name, but for organizational reasons, it's recommended to use the name of the project loaded in the CLP.	'PLC1'	The field is a STRING and accepts alphanumeric characters (letters and numbers) and character "_". It is not allowed to start the STRING with numbers or "_". It accepts 49 characters.
Gateway Address	IP address used at Gateway for communication between MasterTool IEC XE and the PLC.	127.0.0.1	1.0.0.1 to 223.255.255.254
Gateway Port	TCP Port to connect with Gateway.	1217	2 to 65534
Device Name	PLC name shown in the Device Communication Configuration tab. It is the STRING that precedes the hexadecimal value between []. Only enabled when the checkbox Use TCP/IP Blockdriver is not marked.	'0000'	The field is a STRING and accepts any characters, just as the name of the PLC in the device Communication Configuration tab. It accepts 49 characters.
Active IP Address	PLC IP Address. Only enabled when the checkbox Use TCP/IP Blockdriver is marked. Used only	127.0.0.1	1.0.0.1 to 223.255.255.254

	when the configuration is not redundant.		
PLC A IP Address	IP address of PLC A. Only enabled when the configuration is redundant. It is the primary address with which the server will communicate in case there's a failure.	127.0.0.1	1.0.0.1 to 223.255.255.254
PLC B IP Address	IP address of PLC B. Only enabled when the configuration is redundant. It is the secondary address with which the server will communicate in case there's a failure.	127.0.0.1	1.0.0.1 to 223.255.255.254
Device Port	TCP Port. Only enabled when the checkbox <i>Use TCP/IP Blockdriver</i> is marked.	11740	11740 or 11739

Table 8-1. PLC Configuration Parameters for each OPC Server

Notes:

Gateway Address: Gateway configuration is the same as described at **Communication Settings**, because the OPC Server uses the same communication gateway and protocol as the communication between PLC and MasterTool IEC XE. The option *Using the Gateway Embedded in PLC* may be selected when the PLC's own gateway is to be used, to optimize OPC communication when there's more than one Client OPC station connected to the same PLC. This feature avoids excessive data traffic through a particular station.

Device Name: This parameter must receive the name shown by the selected PLC as active in the **Communication Settings** screen and shown at **Ethernet Interfaces**. It is used when checkbox *Use TCP/IP Blockdriver* is not marked.

IP Address: This field must be filled with the IP address of the **Ethernet Interfaces** that are going to be used for communication between MasterTool IEC XE and the PLC. This option is enabled when the checkbox *Use TCP/IP Blockdriver* is marked. This method requires port 11740 to be used.

New Device: This button adds a new PLC to the OPC Server. Below the field *Devices*, a list of all OPC Server configured PLCs is shown. The existing configuration can be edited by selecting the PLC in the list and changing its parameters. The unused PLC configurations can be excluded. The maximum number of configured PLCs in an OPC Server is 16.

Read Project Configuration: This button attributes the opened project configuration to the PLC being edited. The *Address* and *Gateway Port* information are the ones configured as described at **Communication Settings**, according to the Active Path. The *IP Address* are read from NET1 Ethernet interface and the *Device Port* will be 11740. It is important to note that this option requires an opened project with a defined Active Path. If any condition is not met, an error message will be presented and no data will be changed.

NOTE: To store OPC Server configuration, MasterTool IEC XE needs to be executed with administrator privileges in the Operating System. Depending on its version, this right is given at the program's opening. Right-click over MasterTool's icon and select "Run as administrator".

NOTE: PLC configuration in an OPC Server are not stored in the project created with MasterTool IEC XE. For this reason, they can be made with an opened or closed project. The configuration is stored in a configuration file where the OPC Server is installed. When changing the configurations, it is not necessary to load it to the PLC, but depending on the OPC Client, it may be necessary to connect again to the Server or load the configuration for the data to be properly updated.

Symbol Configuration Object

To configure OPC communication, just configure correctly the node and indicate the variables to be used in the communication. There are two ways to indicate which project variables are available at the OPC Server: using the Symbol Configuration environment or using attributes. Both cases require the object Symbol Configuration to be added; Just right-click the Application object and select the Symbol Configuration

Below is a brief description of the fields found in this object, presented in Figure 8-22. For further information on utilization and configuration of Symbol Configuration, as well as OPC communication, consult Nexto Series CPUs User Manual – MU214605, OPC DA section.

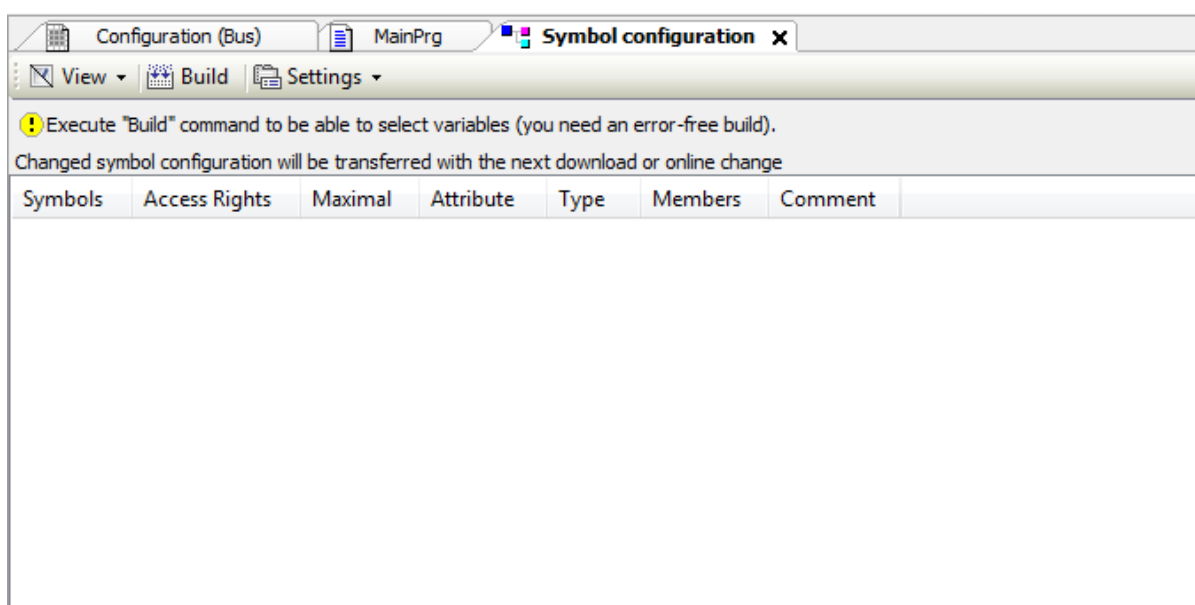





Figure 8-22. Symbol Configuration Object

Table 8-2 presents the description of fields in the symbol configuration screen.

Field	Description
Symbols	Variable identification that will be made available to the OPC Server.
Access Rights	Indicates the possible access level of the declared symbol. When this column is not used, it remains empty the access level is maximum. Otherwise the level can be modified by clicking on the field. The possibilities are: Read only  Write only  Read and write 
Maximal	Indicates the maximal access level that can be attributed to a variable. The symbols have the same meaning as the ones in Access Rights. It's not possible to change and it is indicated by the presence of {attribute 'symbol'}.




Attribute	Indicates whether {attribute 'symbol'} is being used when the variable is declared. When not used, this column remains empty. When used, the behavior is the following: {attribute 'symbol' := 'read'} column shows  {attribute 'symbol' := 'write'} column shows  {attribute 'symbol' := 'readwrite'} column shows 
Type	Data type of the declared variable.
Members	A button is enabled in this column when the data type is a Struct. By clicking it, it's possible to select which structure elements are available to the OPC Server.
Comment	Comment of the variable inserted in the POU or GVL where it's declared. To appear as a comment, it must be placed one line above the declaration when in textual mode, or in the comment column when using tabular mode.

Table 8-2. Symbol Configuration Screen Fields Description

Testing OPC Communication Using Simulation

MasterTool IEC XE has a feature to connect the simulator to the OPC Server. This way, it is possible to test communication with a SCADA system without an actual PLC.

The project's configuration is just as shown before; However, PLC configuration in the OPC Server must be changed. The checkbox *Use TCP/IP Blockdriver* must always be marked. The *IP Address* must always be 127.0.0.1 and the *Device Port* must be 11739. After making these changes, just select the *Simulation* option at the project and make a *Login*.

While simulation is being executed, the simulator's variables will be available to the OPC Server. By disconnecting it, they will no longer be available. This configuration is also only possible if the gateway is configured as localhost.

Only one instance of the simulator with OPC Server can work in a computer at a time.

Limitations described at **Simulation Mode** still apply while using OPC Server.

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.

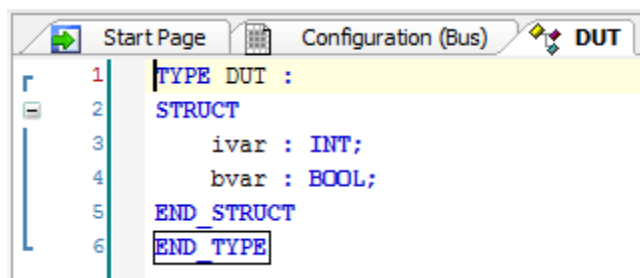


Figure 8-23. DUT Editor Window

FBD/LD/IL Editor

A combined editor is available for editing POU 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.

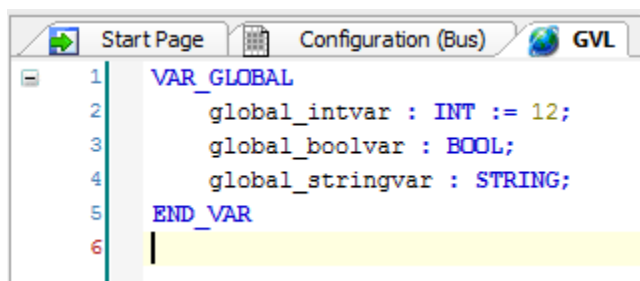


Figure 8-24. GVL Editor

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 POU's 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.

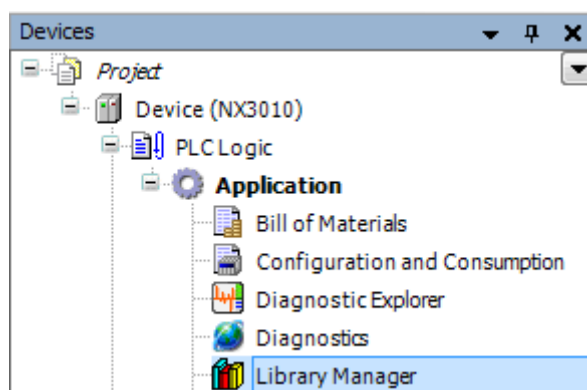


Figure 8-25. Library Manager Object in the POU's View Window

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**.

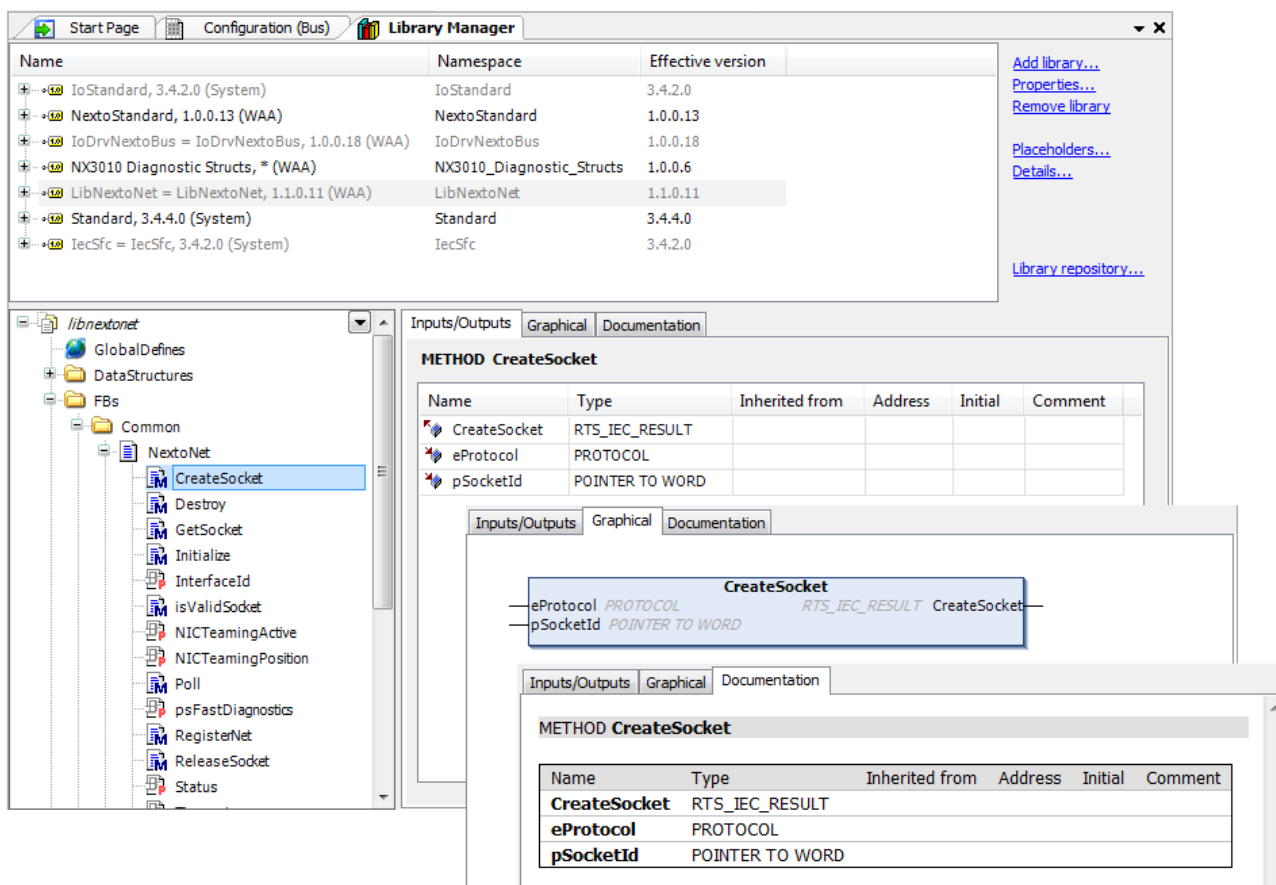


Figure 8-26. Library Manager Editor


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 available in multiple instances in the project. See also **Library Management** (IEC 61131 Programming Manual) for information on library namespaces.
- *Effective Version*: “Effective version” is that library version which will be currently used due to the definition in the library properties.

Libraries, which have been included in a project automatically by a plug-in, are, displayed grey-colored, those that have been added manually (*Add Library*) are displayed in black color.

An icon before the library name indicates the type of the library:

- MasterTool IEC XE library (+). 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.

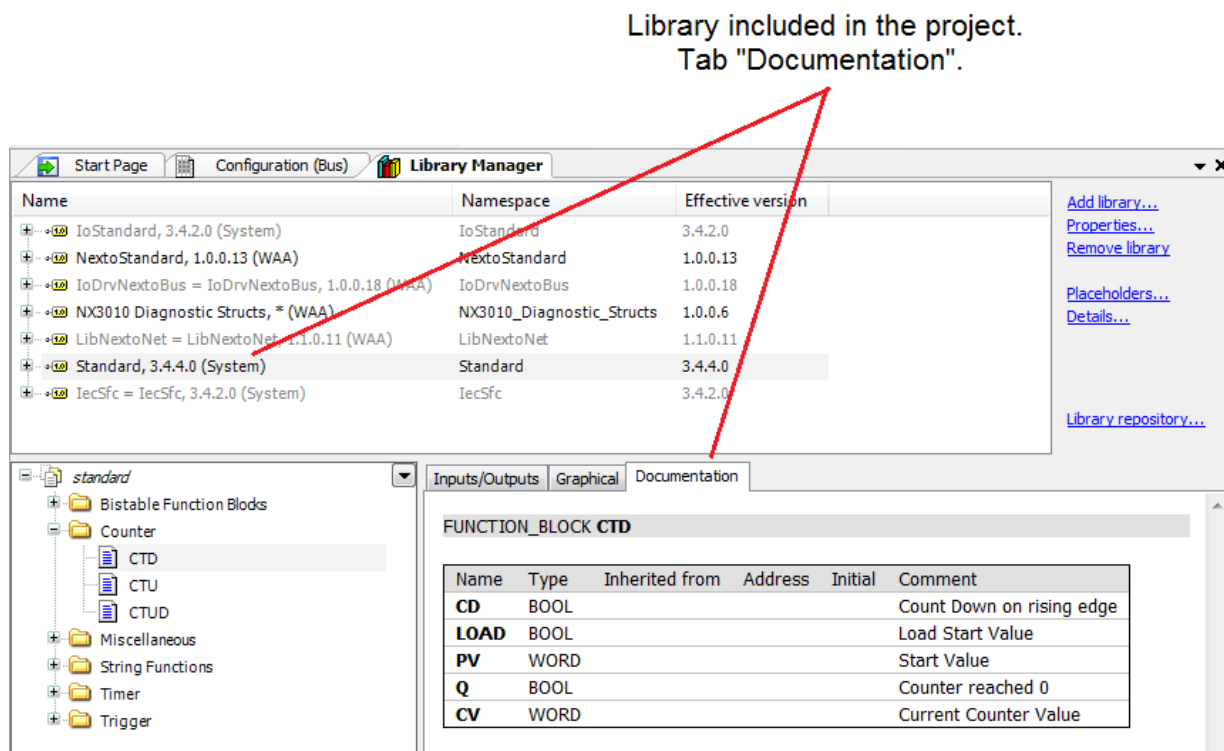


Figure 8-27. Commented Library Modules

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.

Recipe Manager Editor

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

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

Adding the Object

To add the object *Recipe Manager*, open the *Application's* context menu and then *Add Object*, and lastly *Recipe Manager...*, as shown in Figure 8-28.

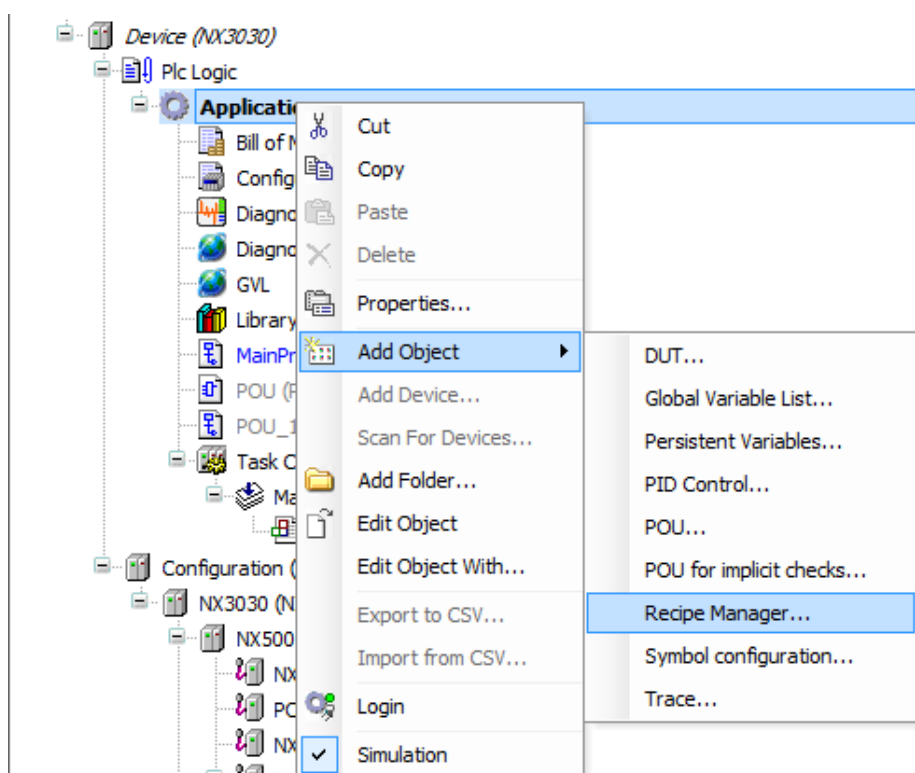


Figure 8-28. Adding the Object Recipe Manager

Then a dialog to confirm the creation of the *Recipe Manager* object will be shown.

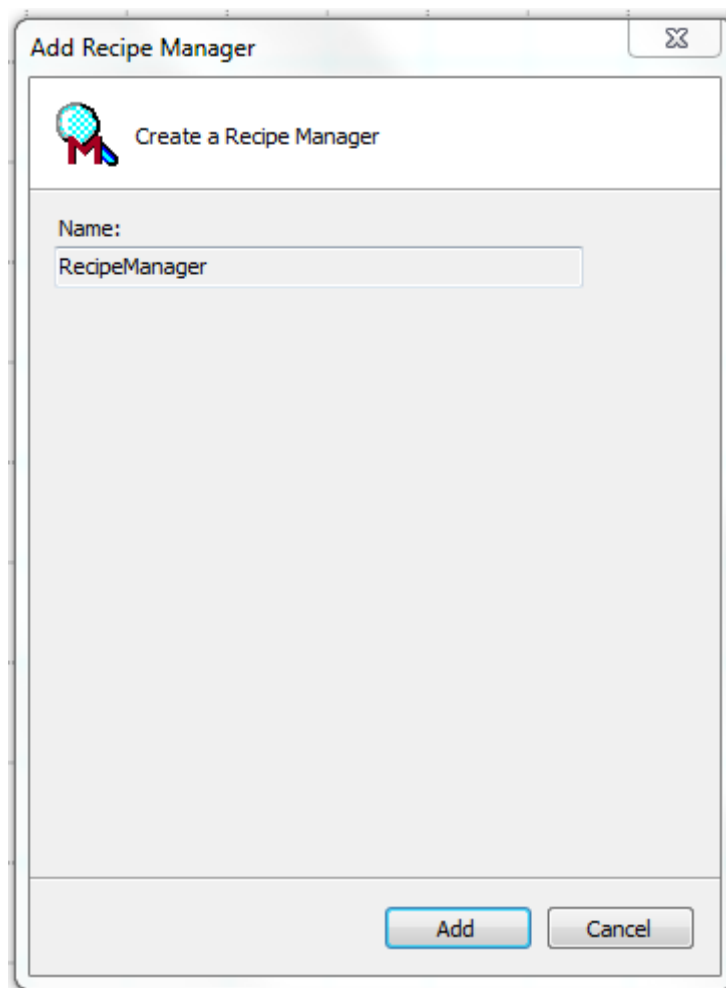


Figure 8-29. Adding the Object Recipe Manager

Right after it is added, its editor will open; It can also be opened by a double click on *Recipe Manager* or by the *Edit Object* command.

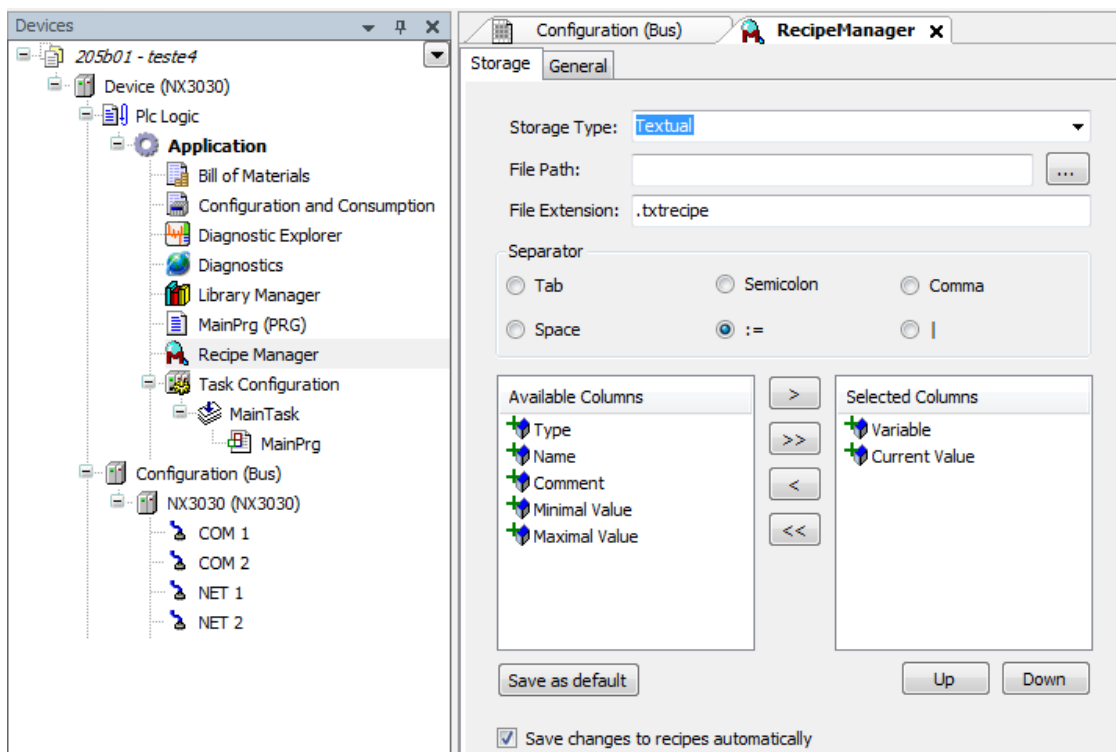


Figure 8-30. Recipe Manager Object Editor

Storage Tab Fields

Recipe Manager editor's *Storage* tab is used to configure the backup file generated by the save command of *Recipe Definition* editor. These configurations are also used when the user read previously saved files.

This screen presents several options to configure the file, described below

ATTENTION:

Files saved with a given configuration e.g. binary format with “:=” separator, are only correctly read if the *Storage* tab configuration matches them at the moment of the reading.

Storage Type

In this selection box, the user chooses between saving the *Recipe* file in textual form or binary form.

File Path

This field is not used.

File Extension

In this text box, the user chooses the extension with which the file is saved. Any letter or number are allowed, but special characters are not.

Separator

In this group, the user selects the way in which the file will separate the information.

Example: “NAME:=TEST:=INT:=10”. Where the separator is “:=”.

Example: “NAME|TEST|INT|10”. Where the separator is “|”.

Selection Lists

Selection Lists enable the user to choose which items will be saved in the *Recipe* backup file, leaving the files one wants to store in the file at the *Selected Columns* list.

< and > buttons are used to insert individual items from one table to the other, while << and >> buttons are used to move all files at once.

Double click over a table entry does the same as buttons < and >, moving from one table to the other.

NOTES:

- It's not possible to move the item *Current Value* from the list *Selected Columns* to the *Available Columns*.
- The *Current Value* in the selection lists refers to the values attributed to the recipes in the *Recipe Definition*, and not the current value in the application.
- To view the selected values in the *Available Columns* and *Selected Columns* lists, it is necessary to click at the top of each field.

The *Up* and *Down* buttons change the order of the items in the list.

ATTENTION:

Files saved with a given items order in the list are only correctly read if the item order in the *Storage* tab have the same order setup.

Save as Default

Button *Save as Default* make the current configuration of the editor the default. This configuration will be used when the *Recipe Manager* is deleted and a new one is added.

Save Changes to Recipes Automatically

This checkbox allows the user to choose if the Recipe files are created or updated in the CPU's memory after the following actions:

- *Login (Online menu)* with application download, updates the CPU's Recipe file with MasterTool's values.
- *Login (Online menu)* with Online Change, doesn't update the Recipes' values at the CPU file.
- *Reset Warm (Online menu)*, updates the CPU's Recipe file with the values of the last complete project download.
- *Reset Cold (Online menu)*, updates the CPU's Recipe file with the values of the last complete project download.
- CPU power reset, updates the CPU's Recipe file with the values of the last complete project download, if the CPU was not restarted erasing the existing application.

General Tab Fields

There's a checkbox at the *General* tab named *Recipe Management at the PLC*. It enables the user to choose between two forms that the Recipe Manager can be used:

- With the checkbox marked, the recipes are sent to the CPU during the project download and it is possible to access them through the application, using the *WriteRecipe* function of the *LibRecipeHandler* library. If the user wants to change values within the Recipes, a new project download is required. This mode is recommended when recipes values are rarely changed, e.g. finished projects.

- With the checkbox unmarked, the recipe is changed within MasterTool IEC XE editor. The recipe is kept within the programmer and the *WriteRecipe* function won't work, because it affects only recipes loaded in the CPU. This mode is recommended during system implementation, where recipes are changed constantly.

If the *Recipe Management at the PLC* is marked, the *Recipe's* configuration is stored within MasterTool IEC XE and at the CPU at the same time. Changes made in MasterTool won't change the file saved in the CPU memory. To update this file, a new project download is necessary.

Example: If a Recipe is excluded from MasterTool's Editor, it can still be written during the program's execution, because it will be intact within the PLC.

It must be evaluated which mode fits better your project, and not to mix them.

Recipe Definition Editor

Recipe Manager works with one or more *Recipe Definition*. A *Recipe Definition* has a list of variables and one or more *Recipes* (set of values) for these variables. Using different recipes, the user can attribute a set of values for a set of variables being used by the PLC, or save these values in *Recipes*, using them as backup.

Add Object

The *Recipe Definition* object is added at the *Recipe Manager* object through the context menu, using the *Add Object* command, and then selecting *Recipe Definition*, as shown in Figure 8-31.

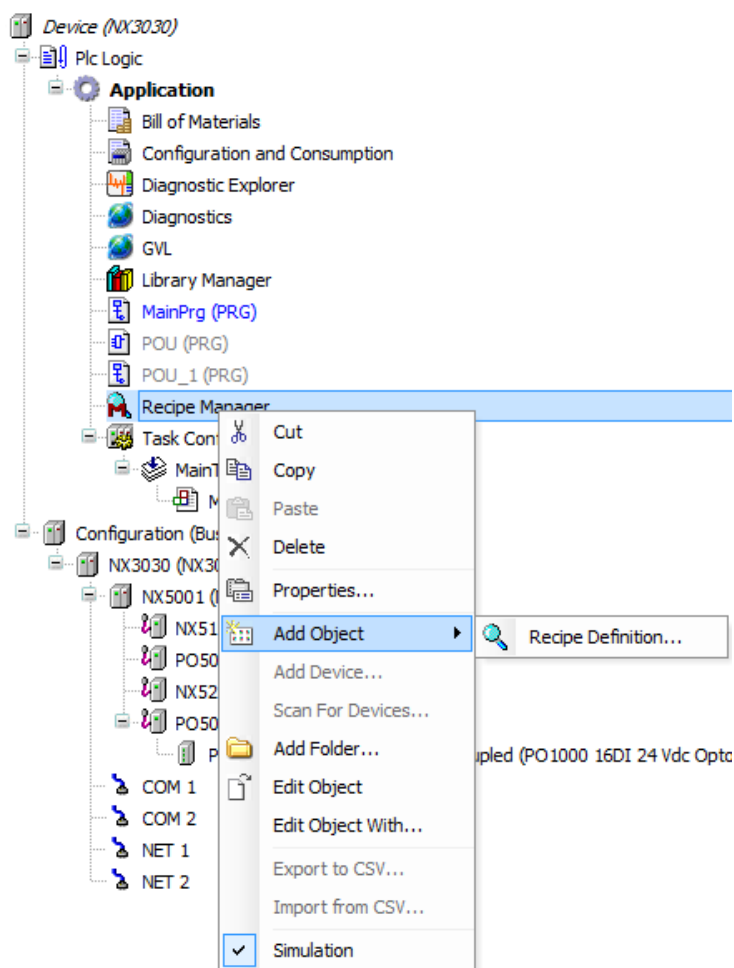


Figure 8-31. Adding a Recipe Definition Object

Then a dialog asking for the *Recipe Definition* name will be opened.

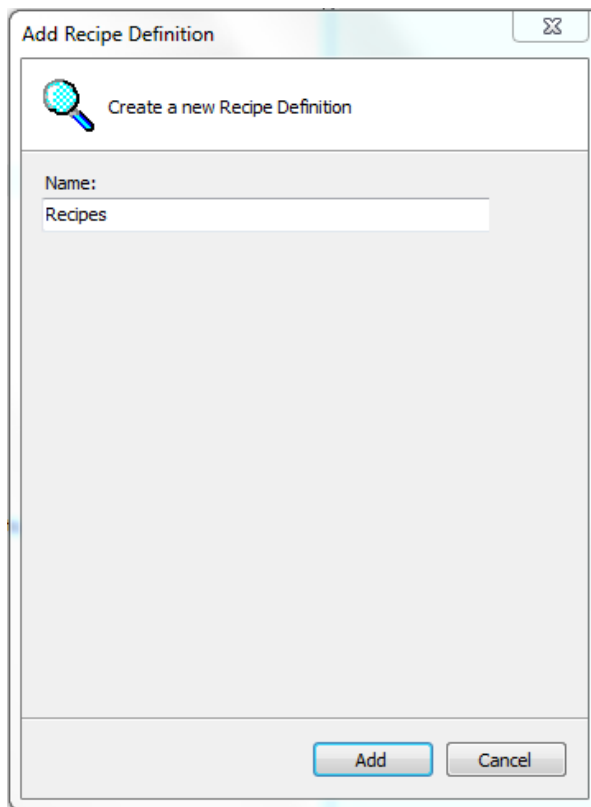


Figure 8-32. Naming the Recipe Definition Object

NOTES:

- The *Name* field cannot start with numbers and it doesn't accept special characters nor spaces.
- There's no character limit for the *Name* field, but it is recommended not to exceed 60 characters.
- There are two limitations:
 - 1) maximum limit of 5 objects of type Recipe Definition, being limited to 32 variables each or
 - 2) a single Recipe with 2000 variables.

The editor opens when the object is created. It can also be opened at any time by a double click in the *Recipe Definition* object or the *Edit Object* command.

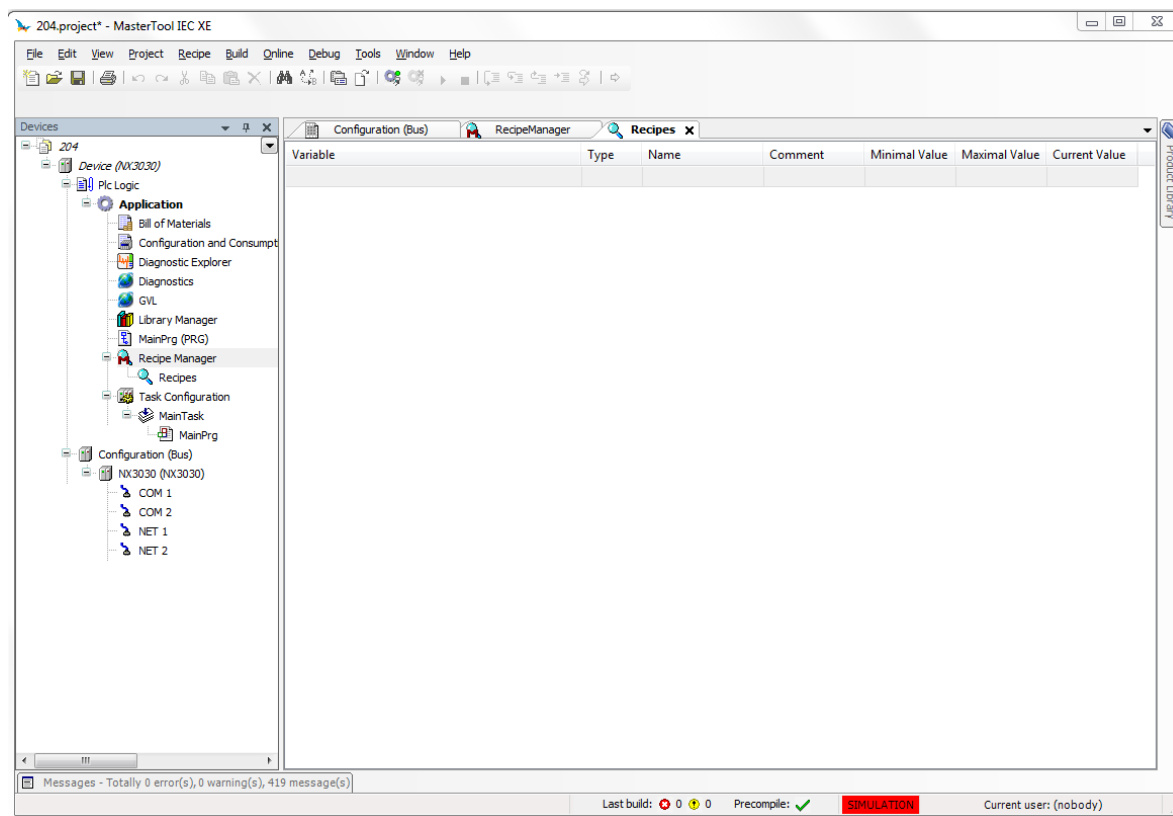


Figure 8-33. Recipe Definition Object Editor

Columns

As seen in Figure 8-33, the *Recipe Definition* object editor has seven columns.

- Variable
- Type
- Name
- Comment
- Minimal Value
- Maximal Value
- Current Value

Each column holds a piece of information about the variables added to the object *Recipe Definition*.

Variable

The *Variable* column holds the name and path of the variable, in the following order: { Path }. { Name }

So if the variable is in the *MainPrg* POU, it will appear in this column as: *MainPrg.VariableName*.

NOTES:

- It's not possible to use Pointers(variables such as POINTER TO <DATA TYPE | FUNCTION BLOCK | PROGRAM | METHOD | FUNCTION>) in *Recipes*.
- It's not possible to use struct variables located in libraries.
- It's not possible to use variables of LTIME type in *Recipes*.
- There are two limitations:
 - 1) maximum limit of 5 objects of type Recipe Definition, being limited to 32 variables each or
 - 2) a single Recipe with 2000 variables.

Type

The *Type* column shows the type of the variable loaded in the previous step.

Name

The *Name* column enables the user to name the variable previously loaded.

Comment

The *Comment* column enables the user to add comments about the variables previously loaded.

NOTES:

- It is not allowed to use characters ‘ e \$ in the *Name* and *Comment* fields. These are reserved characters used to control STRINGS.
- There’s no character limit for the *Name* and *Comment* fields, but it is recommended not to exceed 60 characters.

Maximal Value

The *Maximal Value* column is used to configure the maximal value the variable may receive.

Minimal Value

The *Minimal Value* column is used to configure the minimal value the variable may receive.

NOTES:

- The maximal value a boolean variable may receive is TRUE.
- The minimal value a boolean variable may receive is FALSE.
- If the user configures a minimal value greater than the maximal value, a warning will be generated at the *Messages* window. The behavior cannot be ensured in this situation.

Current Value

This column shows the current value of the variable. It will only appear when in on-line mode.

Actions and Context Menus

There are some context menus that can be used to perform actions in the *Recipe Definition* editor:

- Add a New Recipe
- Remove Recipe
- Load Recipe
- Save Recipe
- Insert Variable

When the user is logged in, more commands are added to the menu:

- Read Recipe
- Read and Save Recipe
- Write Recipe
- Load and Write Recipe

The context menu can be seen in Figure 8-34.

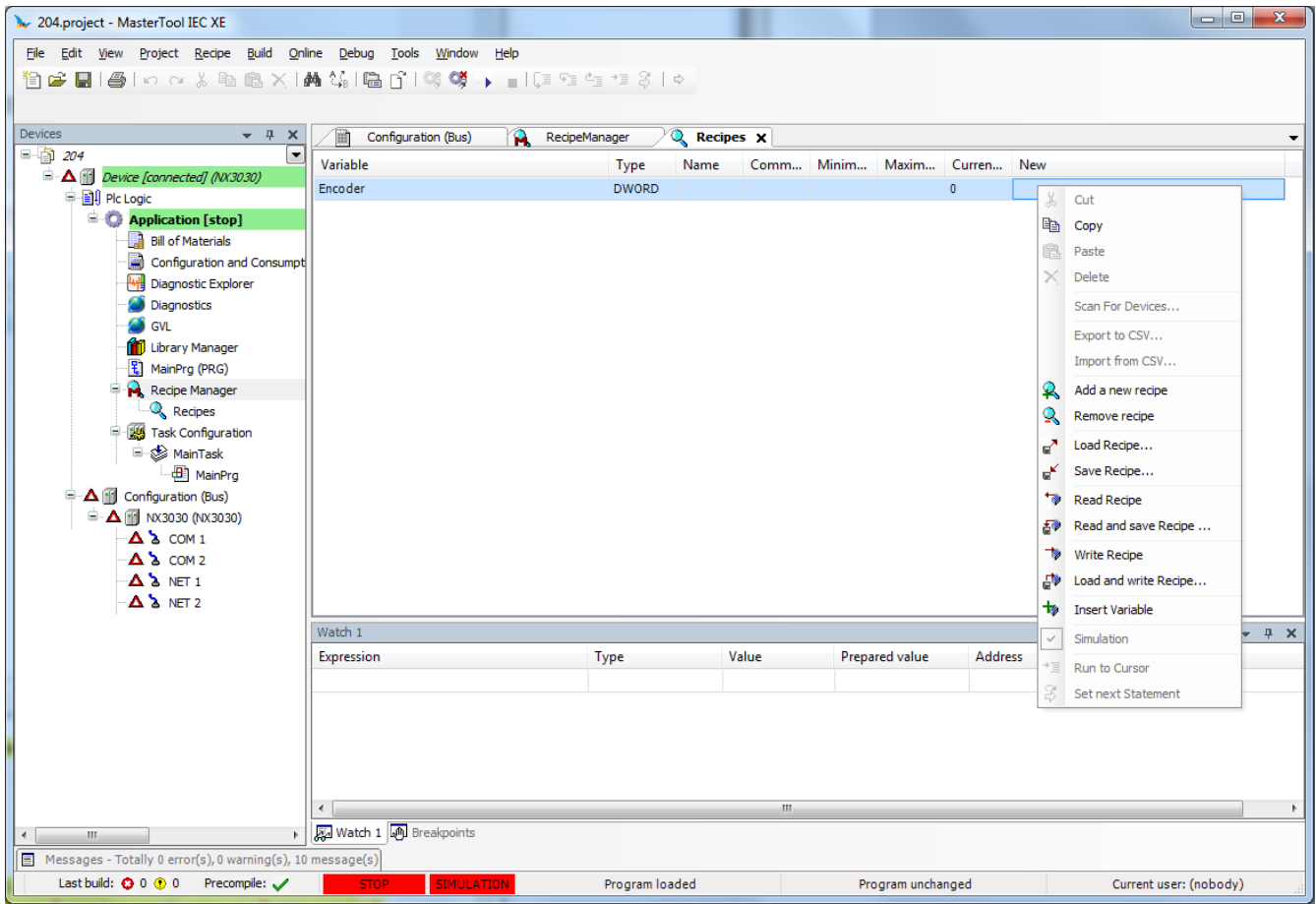


Figure 8-34. Recipe Definition Object Context Menus

Add a New Recipe

Symbol: 

By selecting this command, the window shown in Figure 8-35 opens, requesting the name of the new Recipe. The *Copy from Existing* field enables the creation from an existing Recipe; Just select it on the list, or select <Create Empty>.

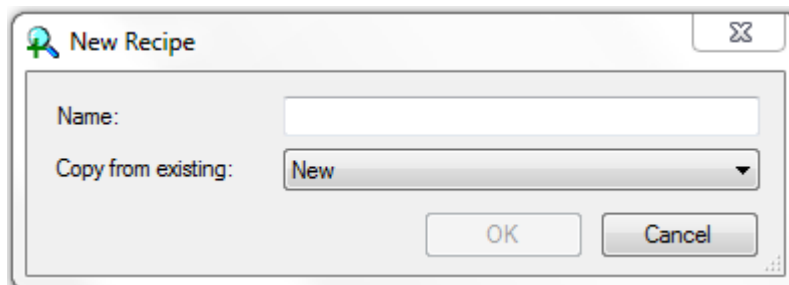


Figure 8-35. New Recipe

Notes:

- It is not allowed to use characters ' e \$ in the *Name* field. These are reserved characters used to control STRINGS.
- There's no character limit for the *Name* field, but it is recommended not to exceed 60 characters.
- There's a 32 *Recipe* limit in each *Recipe Definition*, if this limit is exceeded, a compilation error will be generated.

By adding a new *Recipe*, a new column in the *Recipe Definition* editor will be created. This column will have the chosen name, and its values will be either empty or loaded from a selected *Recipe*.

NOTE: The *Recipe* column lines accept any kind of value, i.e. they are not consisted when added. If there are any inconsistencies between the value added and the data type, compilation errors will be presented at the *Messages* window after using the *Generate Code* command.

Figure 8-36 shows the column added with *New*.

Variable	Type	Name	Comm...	Minim...	Maxim...	Curren...	New
Encoder	DWORD					0	

Figure 8-36. Recipe Definition Editor with New Recipe added

Remove Recipe

Symbol:

This command removes the selected *Recipe* along with all its values.

Load Recipe

Symbol:

This command load *Recipes* previously saved with the *Save Recipe* command. The standard file opening screen is presented.

The *Recipe* file format must have the .txtrecipe extension. The file will either open normally or present one of the following problems:

- File does not contain all values for the *Recipe*'s variables.
- File contains extra values for the *Recipe*'s variables.
- File does not contain all values and contains extra values for the *Recipe*'s variables. The *Recipes* are loaded normally.

NOTE: By executing the *Load Recipe...* command, the values saved in the fields *Variable*, *Type*, *Name* and *Comment* must be identical than the ones in the project. The fields *Minimal Value*, *Maximal Value* and *Current Value* are not consisted and may be different. Only *Recipe* values are loaded.

File Does Not Contain All Values for the Recipe's Variables

It shows a dialog informing that the file doesn't have all the values for all the *Recipe's* variables, and then shows the variables that will have their values lost. It will ask the user if the operation should proceed.

File Contains Extra Values for the *Recipe's* Variables

It shows a dialog informing that the file have values for variables that are not part of the *Recipe*, and then shows the variables that will have their values ignored. It will ask the user if the operation should proceed.

File Does Not Contain All Values and Contains Extra Values for the *Recipe's* Variables

It shows the dialog explained at **File Does Not Contain All Values for the Recipe's Variables** then the dialog explained at **File Contains Extra Values for the Recipe's Variables**

.

Save Recipe

Symbol: 

This command may be used to backup added *Recipes*, creating a .txtrecipe file. The command is used by selecting the *Recipe* and clicking in the save command.

NOTE: It is not possible to save a Recipe with variables of types TIME, TIME_OF_DAY, TOD, DATE, DATE_AND_TIME and DT without a defined value (empty). An error will be presented after the save command and the recipe won't be saved.

Insert Variable

Symbol: 

The *Insert Variable* command creates a new line in the variables list of the *Recipe Definition* editor, where the user can add variables.

Read Recipe

Symbol: 

The user must be logged in to use this command.

The user has to choose the *Recipe* in which the variables will be stored and then select the command. Afterwards, all variable values of *Current Value* column will be added to the respective lines of the chosen *Recipe*.

Read and Save Recipe

Symbol: 

The user must be logged in to use this command.

This command is the union of commands *Read Recipe* and *Save Recipe*. The *Recipe* reads the variables *Current Value* and adds them to their respective line, then opens the standard dialog to save a .txtrecipe backup file.

Write Recipe

Symbol: 

The user must be logged in to use this command.

This command writes the values contained in the lines of the chosen *Recipe* to the variables' *Current Value*.

NOTE: If the value to be written in the PLC is out of the bounds defined in the Recipe, the value written will be the limit itself (maximal or minimal value).

Load and Write Recipe

Symbol: 

The user must be logged in to use this command.

This command is the union of commands *Load Recipe* and *Write Recipe*. A standard dialog will be shown to select the .txtrecipe file, which will be loaded and added to the chosen *Recipe*. Afterwards, the values will be written in the respective lines of *Current Value* at the *Recipe Definition* object.

NOTE: If the value to be written in the PLC is out of the bounds defined in the Recipe, the value written will be the limit itself (maximal or minimal value).


NOTE: The values can be written by a PLC application using function WriteRecipe of the LibRecipeHandler library. For further information, consult MP399609.

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.

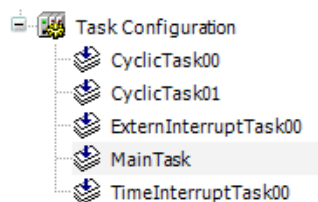


Figure 8-37. Task Configuration in Devices Tree Below an Application

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.

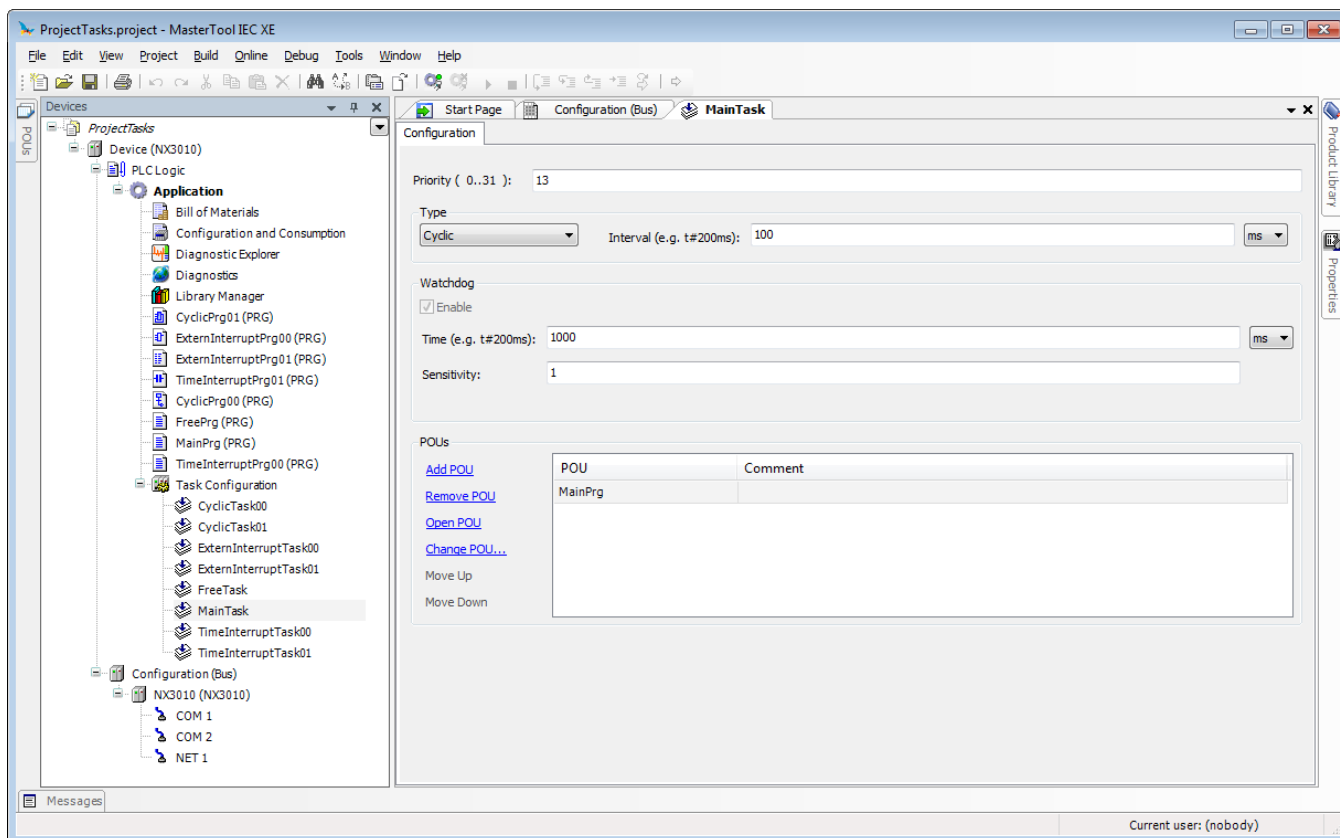


Figure 8-38. Task Editor, Dialog for Configuration of a Task

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 8-39 shows the *Properties* window of a CPU NX3010.

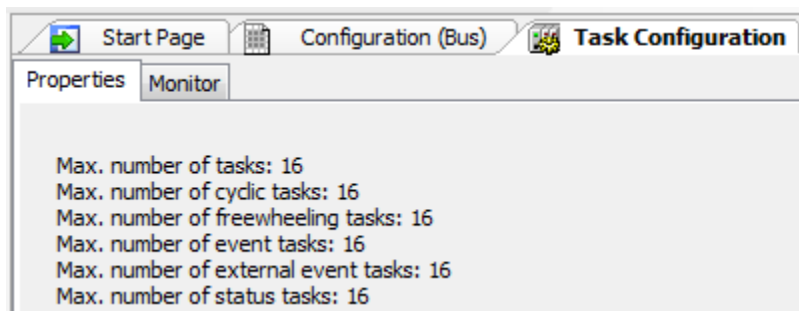


Figure 8-39. Task Configuration Properties Dialog

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.

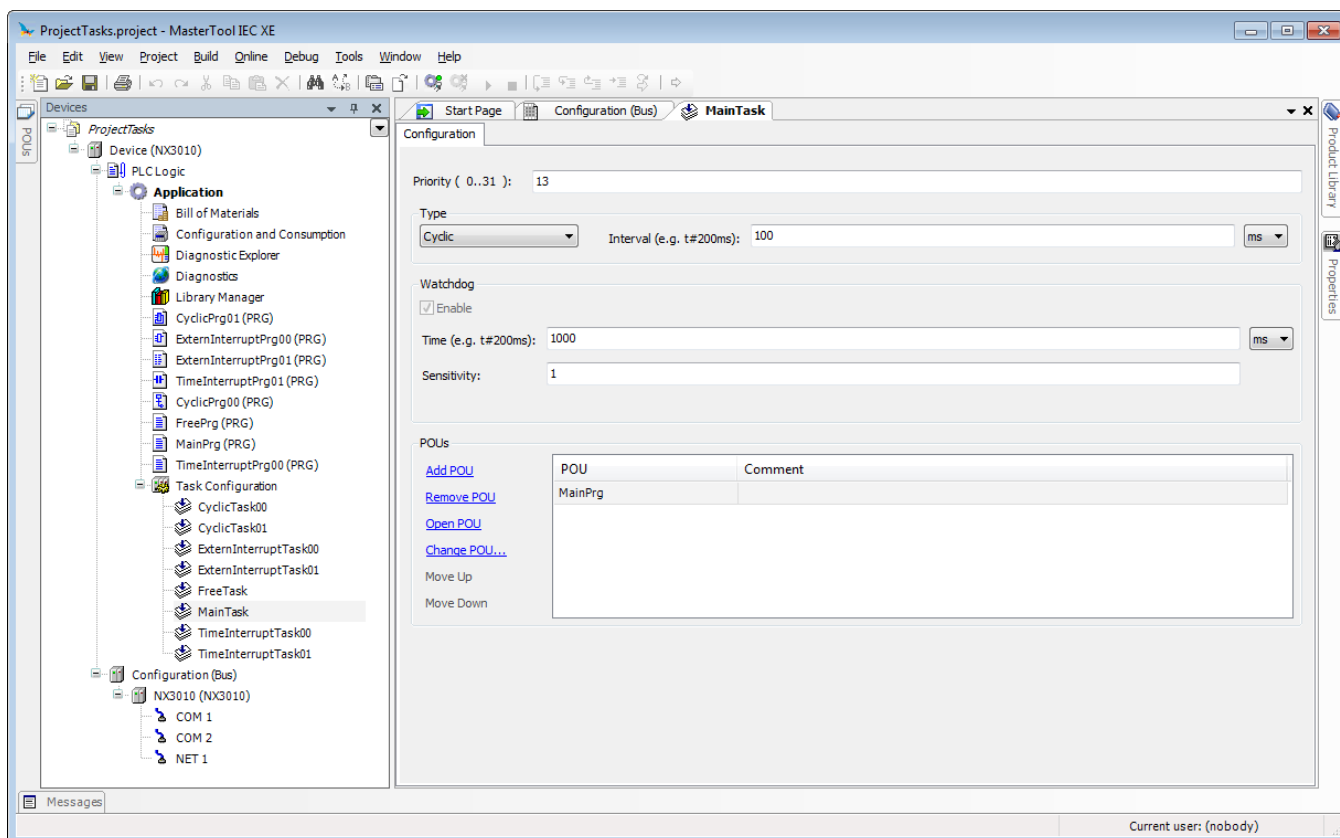


Figure 8-40. Configuration Dialog for a Task

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.

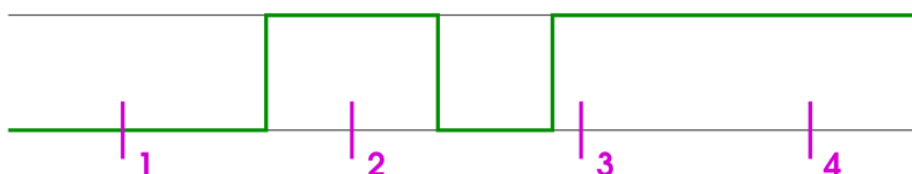


Figure 8-41. Resulting Behavior of the Task in Reaction to an Event (Green Line)

At sampling points 1-4 (magenta) tasks of different types show different reaction: (Figure 8-41).

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

Table 8-3. 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 8-4. 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's, 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

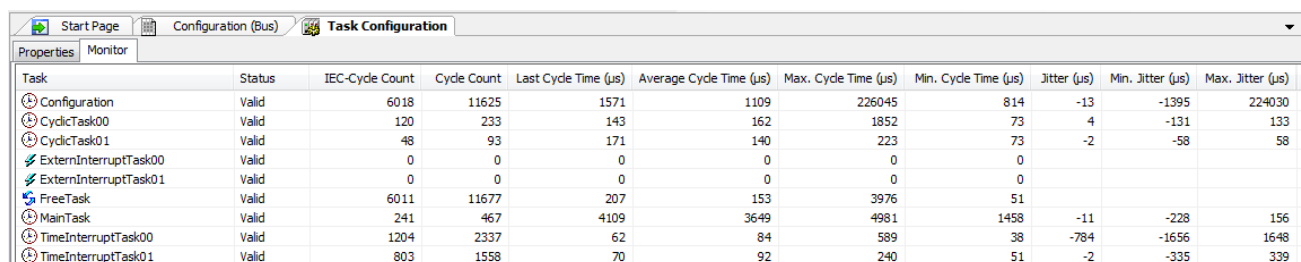
Which task is being Processed?

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 8-42. 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 8-5. 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 Trace
- Stop Trace
- Reset Trigger
- Cursor
- 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.

Start Trace

Symbol: 

This command starts the Trace (the variable values will be shown from the last moment).

If the Trace is started, this command will be disabled.

If the Trace is stopped, this command will be enabled.

Stop Trace

Symbol: 

This command terminates the Trace, i.e. the display will freeze the chart as it was last updated.

If the Trace is stopped, this command will be disabled.

If the Trace is started, this command will be enabled.

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 the option *Mouse Zooming* is activated.

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**.

This command is only enabled while the *Multichannel* command is disabled.

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* command is affecting the x-axis of all diagrams simultaneously.

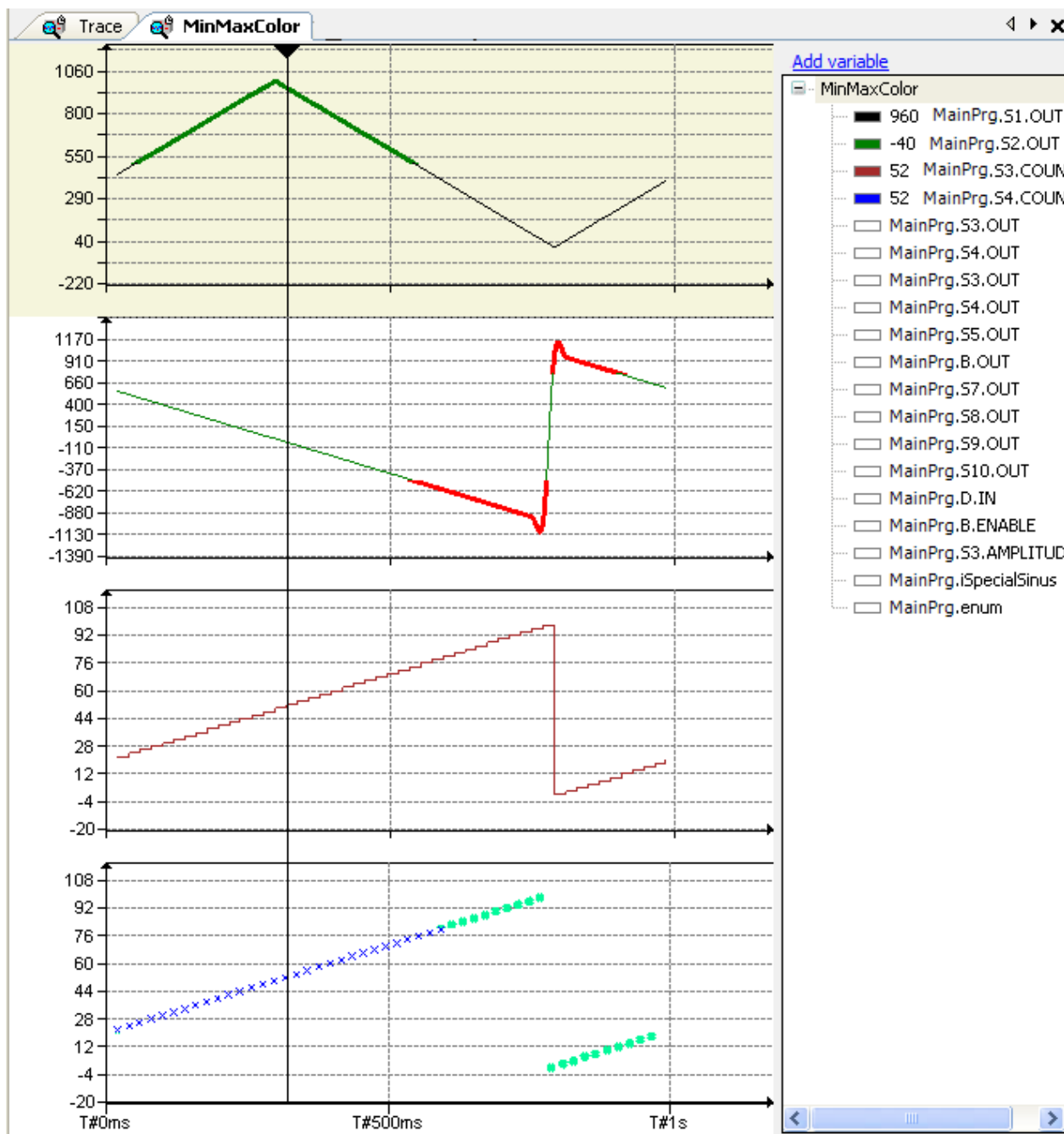


Figure 8-43. Multi-Channel

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 8-6. 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 8-7. 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.

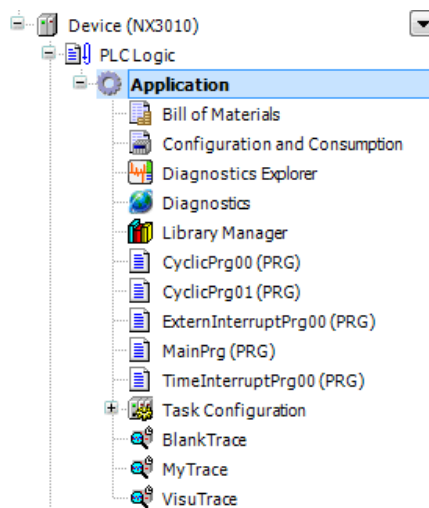


Figure 8-44. Project with Traces

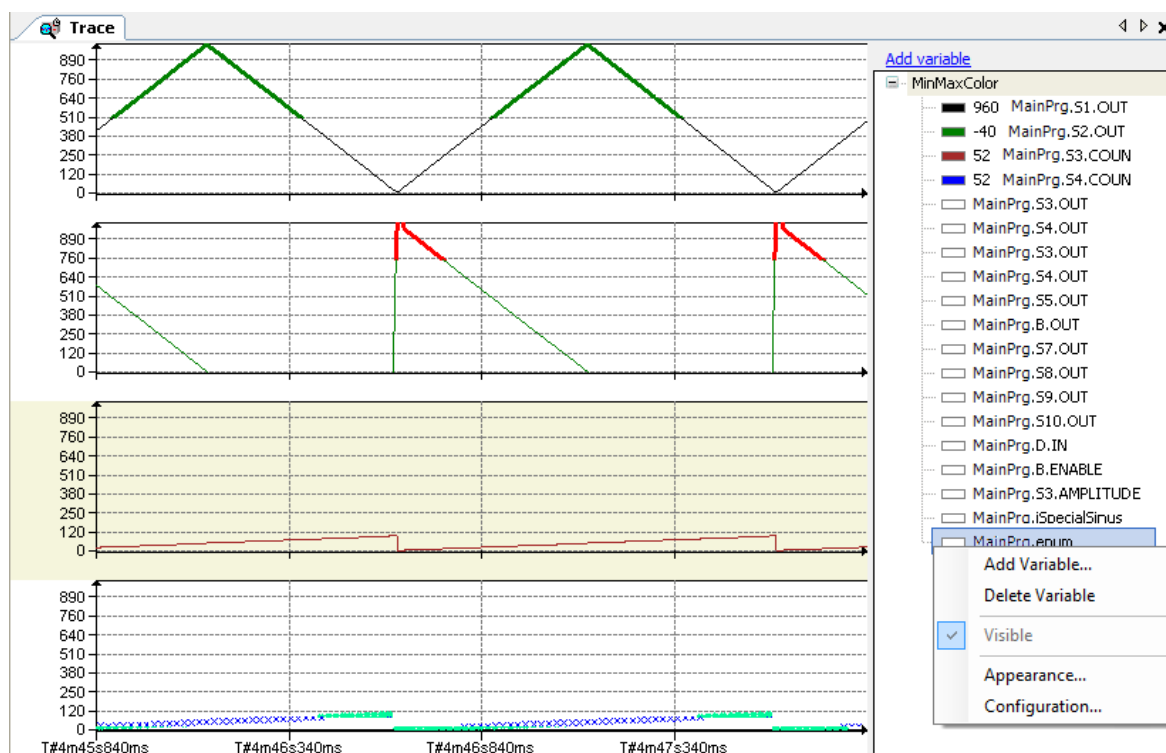


Figure 8-45. Trace Editor Window, in Online Mode

The Figure 8-45 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 mouse scrolling 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.

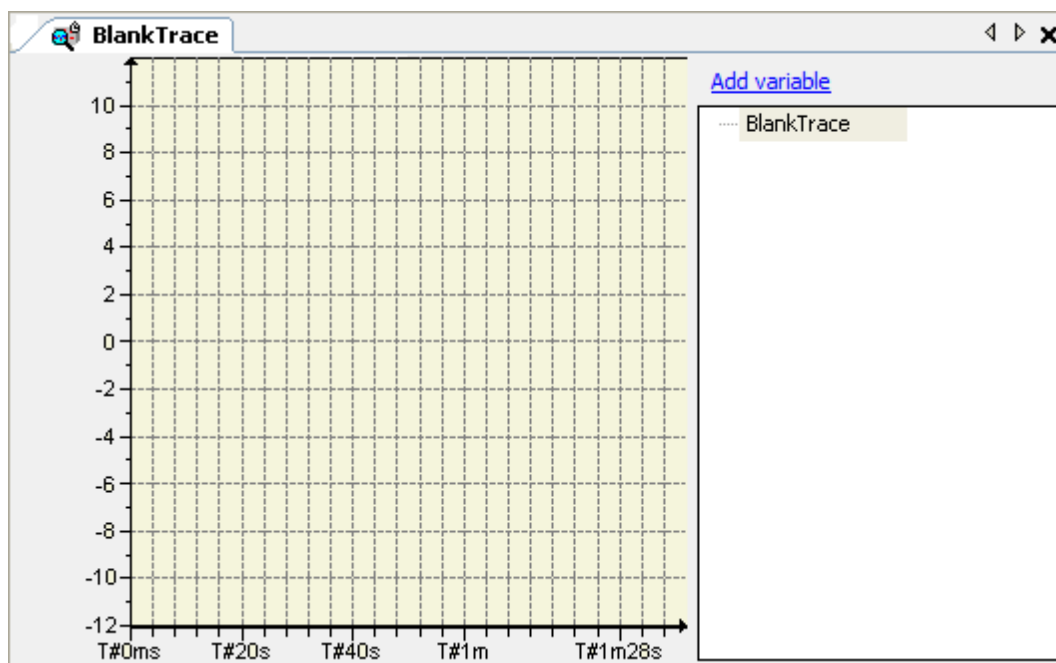


Figure 8-46. Trace Editor Window, No Configuration

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.

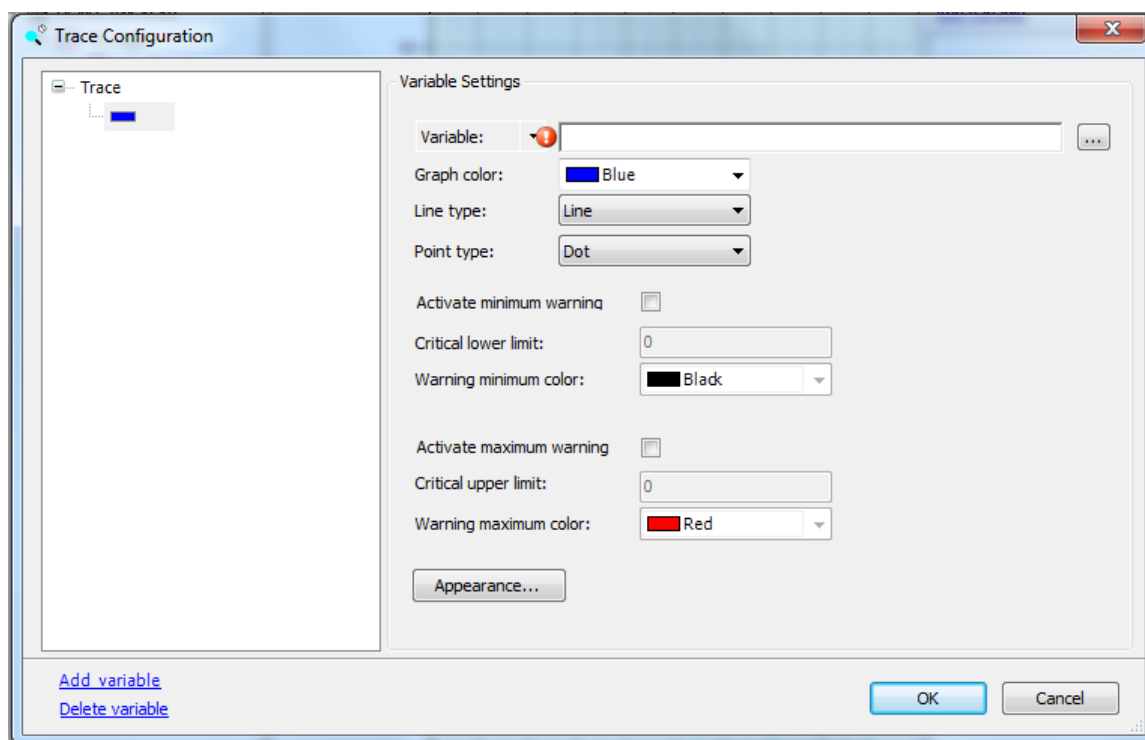


Figure 8-47. Trace Configuration Dialog

- *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.
- *Line Type*: This field specifies the type of line that will be used in the Trace curve. The “Line” option is recommended for large volumes of data. Table 8-8 describes the possible options for the field.

Type	Description
Line	The points are connected through a line.
Step	The points are connected in a step shaped chart, i.e. a vertical line until the Y value of the next point, and from there a horizontal line until the next X value.
None	No graphical display.

Table 8-8. Line Type

- *Point Type*: This field specifies how the points will be displayed in the Trace curve. Table 8-9 describes the possible options for the field.

Type	Description
Dot	The points are represented by dots.
Cross	The points are represented by crosses.
None	No graphical display.

Table 8-9. Point 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.

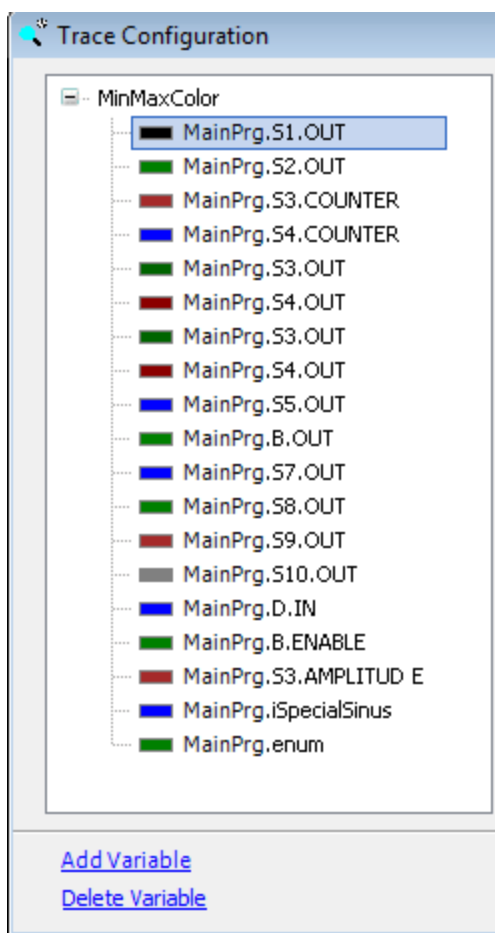


Figure 8-48. Trace Tree in the Trace Configuration Dialog

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.

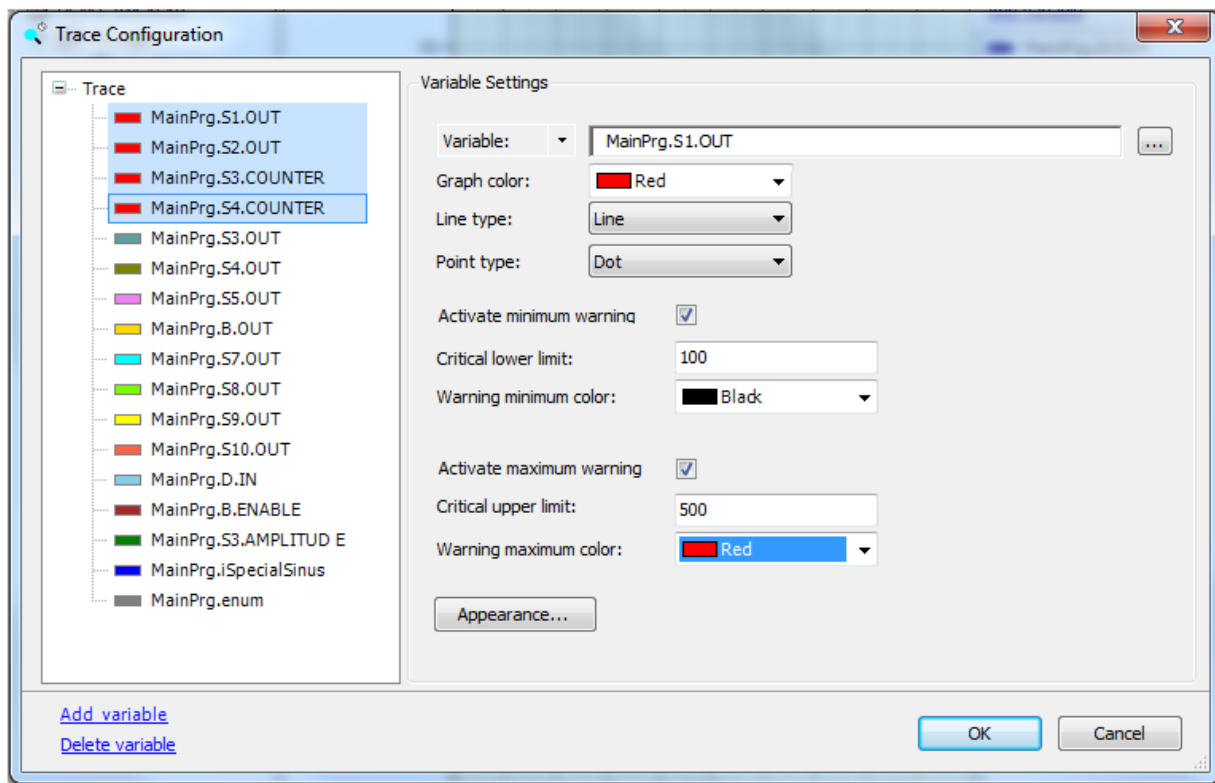


Figure 8-49. Multi Selection of Variables

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.

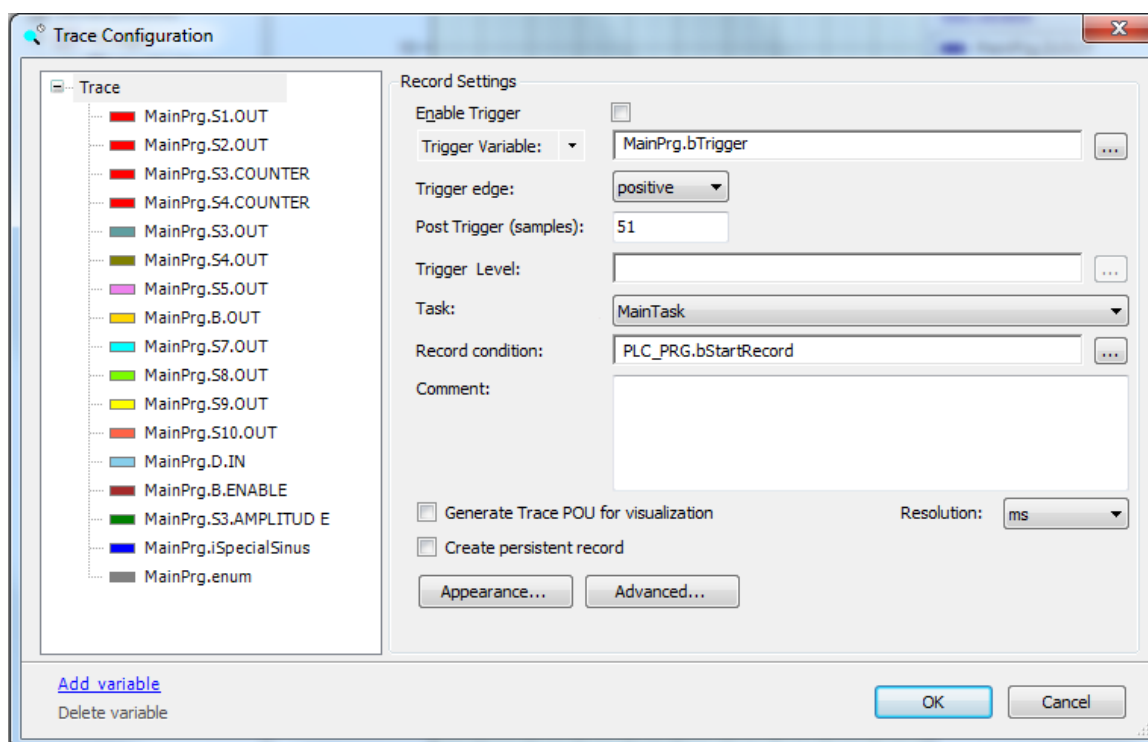



Figure 8-50. Trace Configuration, Record Settings

- *Enable Trigger*: This option enables or disables the trigger system.
- *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 (*Post-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 8-10.

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 8-10. 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*.

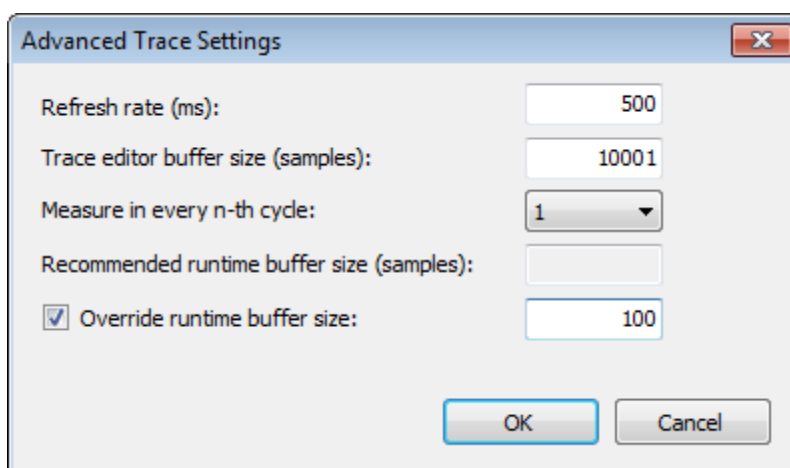


Figure 8-51 Trace Advanced Configuration

- *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.

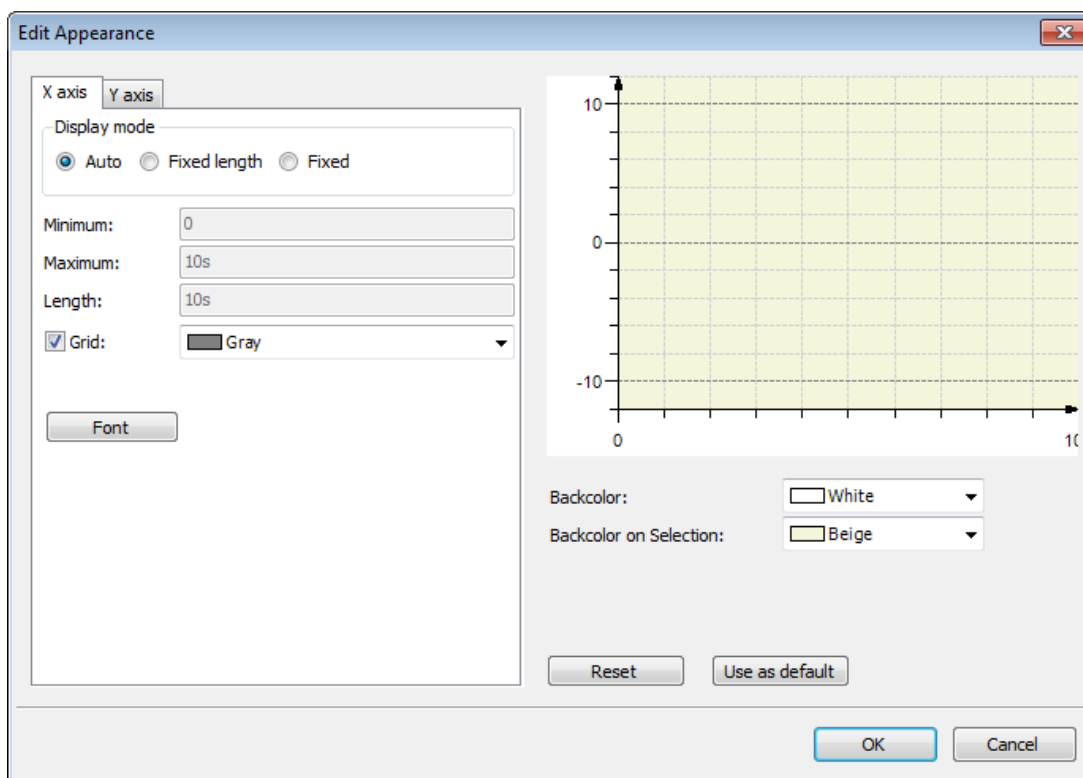


Figure 8-52. Edit Appearance Dialog

- *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.

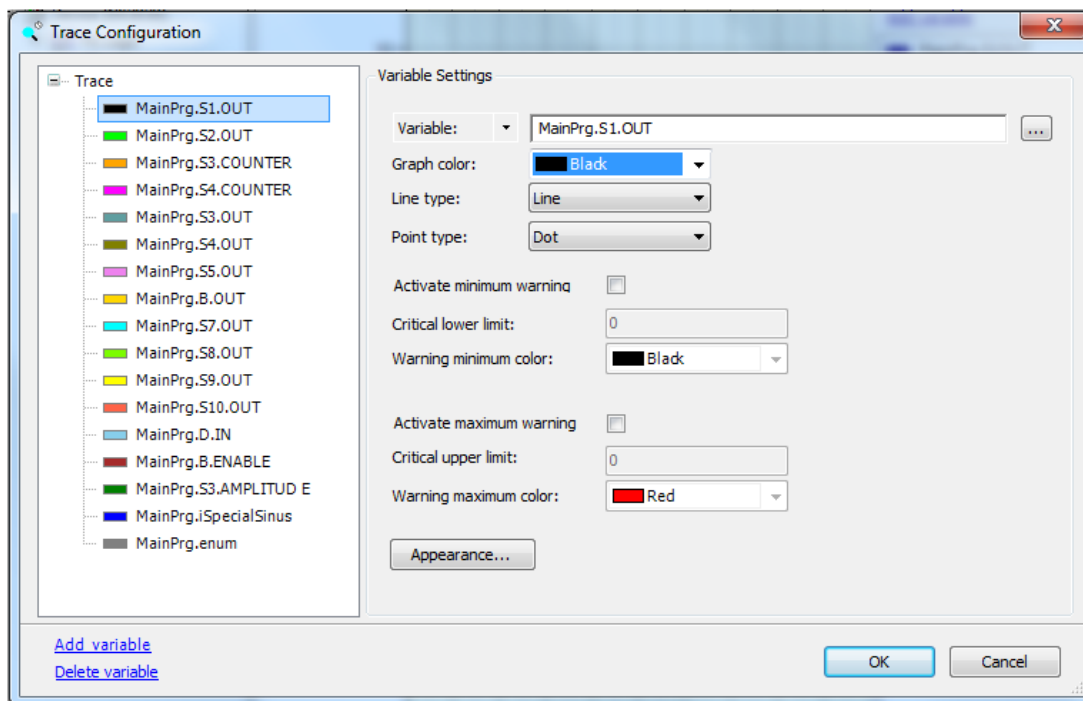


Figure 8-53. Trace Configuration, Variable Settings

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.
- *Line Type:* This field specifies the type of line that will be used in the Trace curve. The "Line" option is recommended for large volumes of data. Table 8-8 describes the possible options for the field.
- *Point Type:* This field specifies how the points will be displayed in the Trace curve. Table 8-9 describes the possible options for the field.
- *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.

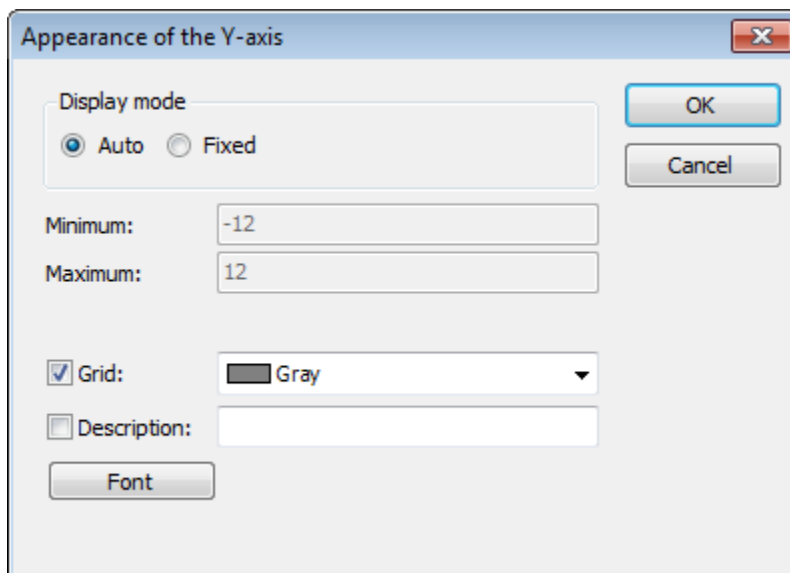


Figure 8-54. Edit Appearance of the Y-Axis Dialog

- *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.

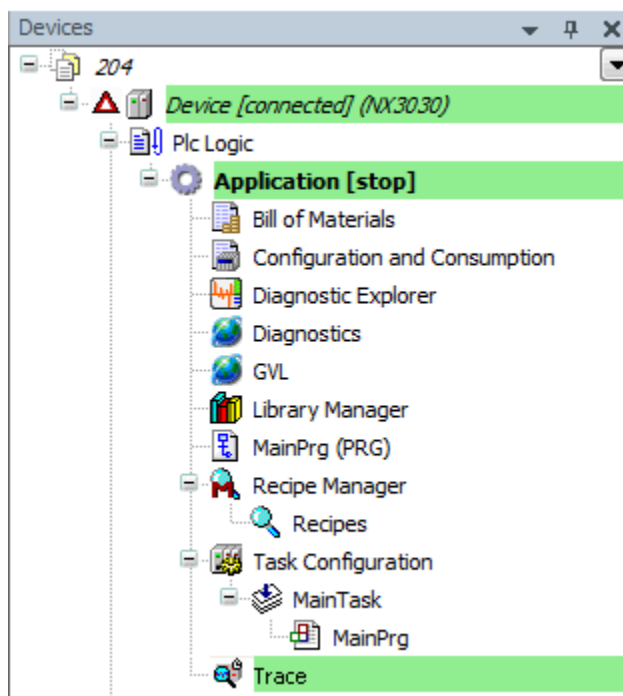


Figure 8-55. Active Trace in Devices Tree

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.

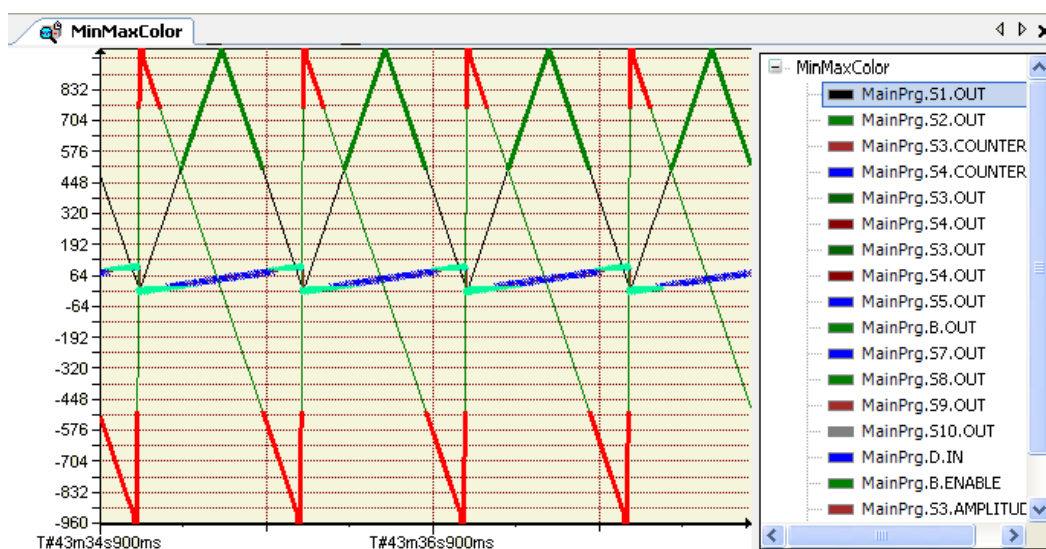


Figure 8-56. Trace Window in Online Mode, Example in Single Channel Mode

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.

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 Watch List

To set up a Watch <n> list in the watch view perform a mouse-click in a field of the Expression column to open an edit frame. Enter the complete path for the desired watch expression.

Syntax for watch expression:

<device name>.<application name>.<object name>.<variable name>.

Example: “Device.Application.MainPrg.ivar” (as shown in Figure 8-57).

The type of the variable will be indicated by an icon:  = input,  = output and  = normal. If a comment has been added to the declaration of a variable, it will be displaced in column Comment.

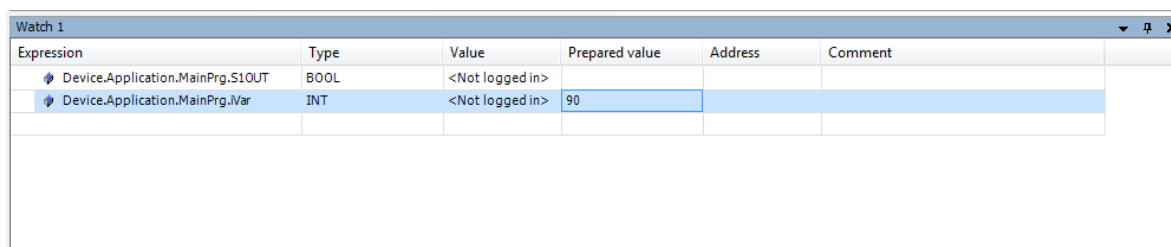
After having closed the edit frame, the *Type* will be added automatically in column Type, and the *Address* will be filled in case the variable is attributed to an address.

The *Value* column will be used in online mode for displaying the current value of the expression.

To appoint a prepared value to a variable you may click in the assigned field of the column *Prepared Value* and directly type in the desired value.

In case of a boolean variable the handling is even easier. You can toggle boolean preparation values by use of the <ENTER> or <SPACE> key according to the following order: If the value is TRUE, the preparation steps are FALSE -> TRUE -> -> no entry. If the value is FALSE, the preparation steps are TRUE -> FALSE -> no entry.

Do the same for the desired further expressions/variables in further lines. See an example in the next picture, which shows the watch view in offline mode. Notice that in case of a structured variable, like here the function block instance, the particular instance components automatically get added when you enter the instance name (see in the example: “Device.Application.MainPrg.fbinst”). They can be displayed/hidden in a fold via a mouse action on the plus-/minus sign.





Expression	Type	Value	Prepared value	Address	Comment
 Device.Application.MainPrg.S1OUT	BOOL	<Not logged in>			
 Device.Application.MainPrg.IVar	INT	<Not logged in>	90		

Figure 8-57. Watch View in Offline Mode

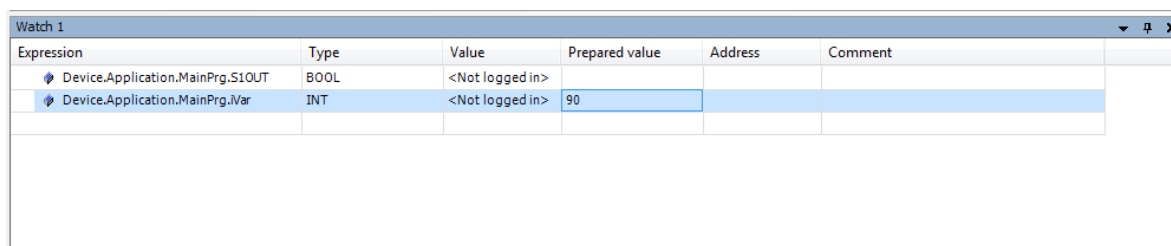
In online mode the view can be used for monitoring.

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 Watch List**.





Expression	Type	Value	Prepared value	Address	Comment
 Device.Application.MainPrg.S1OUT	BOOL	<Not logged in>			
 Device.Application.MainPrg.IVar	INT	<Not logged in>	90		

Figure 8-58. 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.*

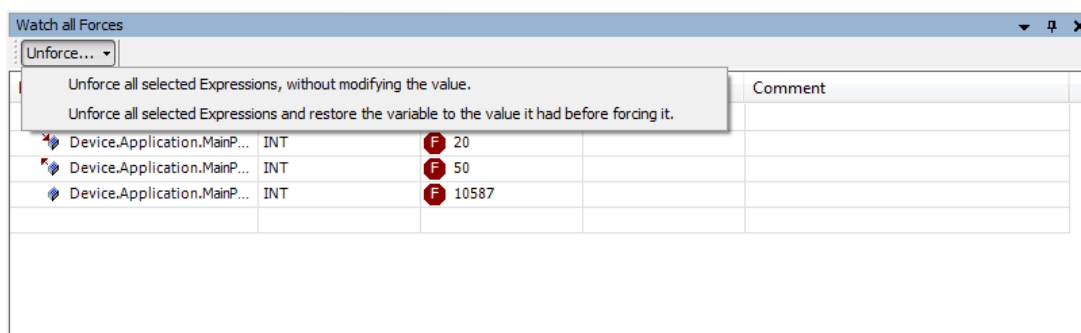


Figure 8-59. Watch All Forces

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 CPU version 1.3.0.20 or later is necessary.

ATTENTION:

When copying a MODBUS device by direct representation from one project to another, the addresses of the IEC variables are not taken together, so they must be configured again.

MODBUS RTU Master for Direct Representation (%Q)

By adding the MODBUS RTU Master for Direct Representation (%Q), the following screen will appear.

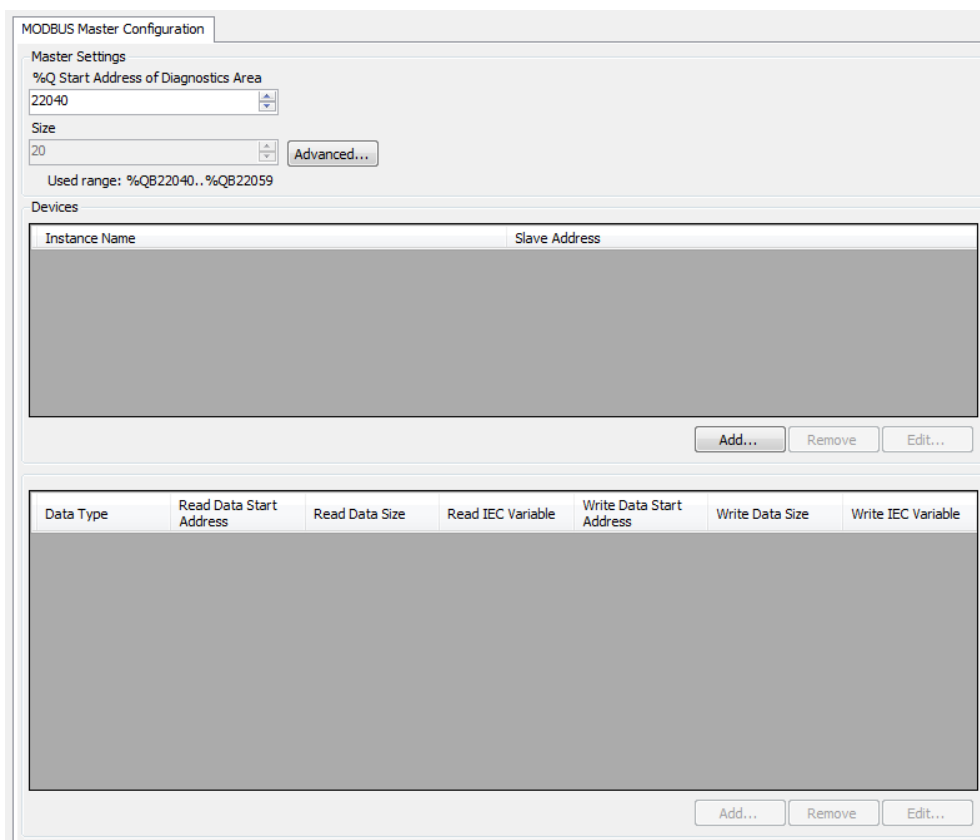


Figure 8-60. MODBUS RTU Master for Direct Representation Screen

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 8-11. Initial Address for Diagnosis on MODBUS Master for Direct Representation

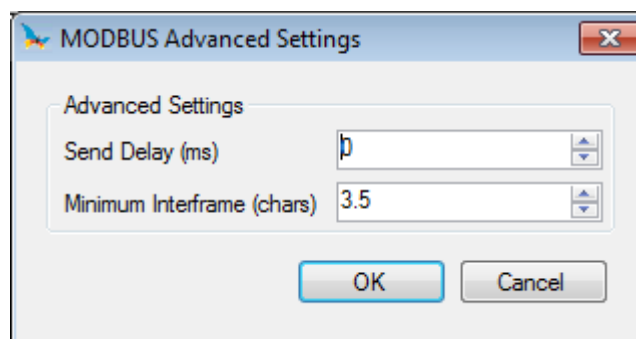


Figure 8-61. MODBUS Master for Direct Representation Advanced Settings

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 8-12. Timings Setting on MODBUS Master for Direct Representation

Adding a Device

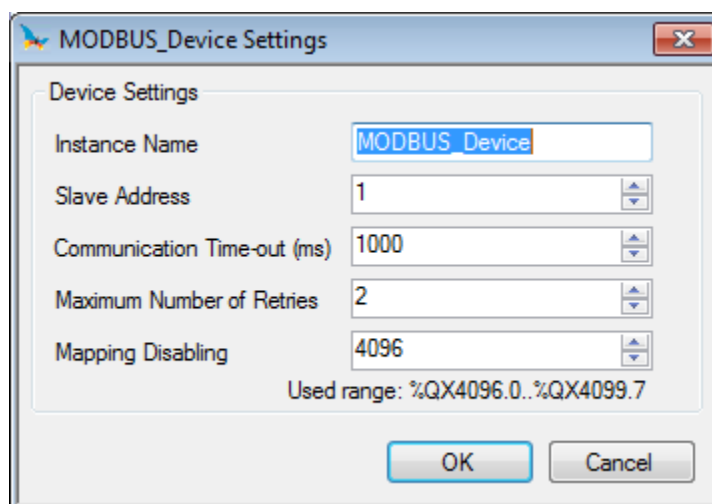


Figure 8-62. Adding a Device on MODBUS Master for Direct Representation

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 8-13. 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.

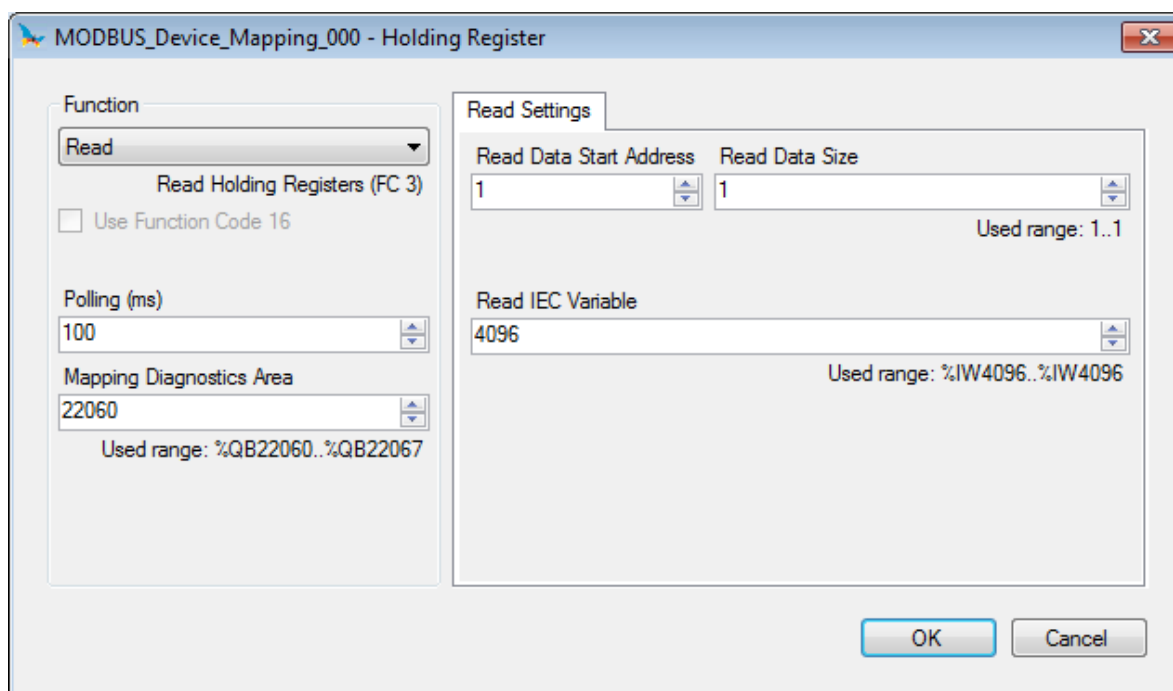


Figure 8-63. MODBUS Relation

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 8-14. MODBUS Master for Direct Representation Relation

MODBUS RTU Master for Symbolic Mapping

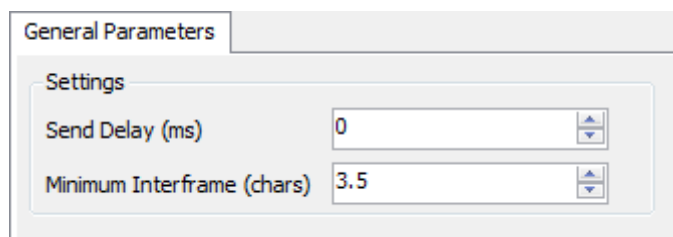


Figure 8-64. MODBUS RTU Master for Symbolic Mapping Screen

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 8-15. 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 8-65, follow the parameters below:

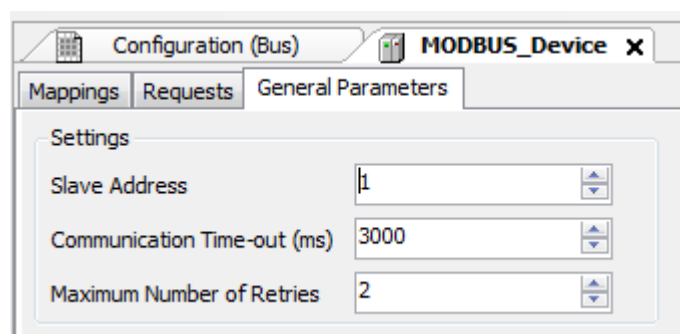


Figure 8-65. Device General Parameters on MODBUS RTU Master for Symbolic Mapping

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 8-16. Device Configurations on MODBUS RTU Master for Symbolic Mapping

Mapping Configuration

The mapping configuration on MODBUS RTU Master for Symbolic Mapping, seen on Figure 8-66, follow the parameters described on Table 8-17:

Figure 8-66. 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 – Write (1 bit) Coil – Read (1 bit) Holding Register - Write (16 bits) Holding Register – Read (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 8-17 Mapping Configurations on MODBUS RTU Master for Symbolic Mapping

Request Configuration

The request configuration on MODBUS RTU Master for Symbolic Mapping, seen on Figure 8-67, follow the parameters described on Table 8-18:

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: INMODBUS_DIAGNOSTIC_STRUCTS.T_DIAG_MODBUS_RTU_MAPPING_1

Generate Diagnostics Variables Generate Disabling Variables

Figure 8-67. MODBUS RTU Master for Symbolic Mapping Data Request Screen

Configuration	Description	Default	Possibilities
Function Code	MODBUS function type	-	01 – Read Coils 02 – Read Input Status 03 – Read Holding Registers 04 – Read Input Registers 05 – Write Single Coil 06 – Write Single Register 15 – Write Multiple Coils 16 – Write Multiple Registers 22 – Mask Write Register 23 – Read/Write Multiple Registers
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 8-18. 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

Slave Address: 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 8-68. 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 8-19. 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.

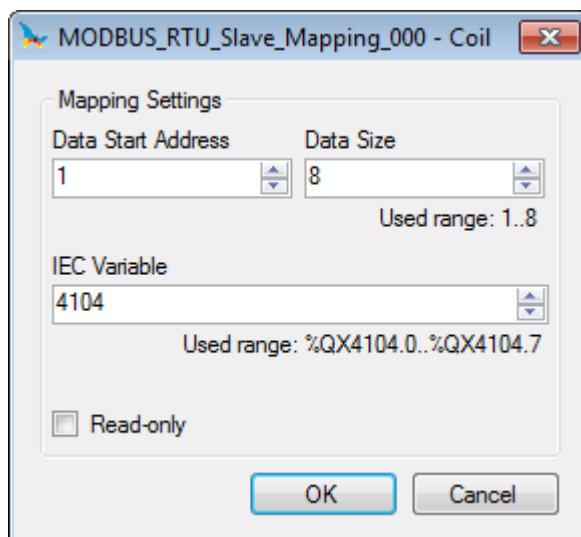


Figure 8-69. Adding a Relation on MODBUS RTU Slave for Direct Representation

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 8-20. MODBUS RTU Slave for Direct Representation Relation Configuration

MODBUS RTU Slave for Symbolic Mapping

By adding a MODBUS RTU Slave for Symbolic 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 8-70.

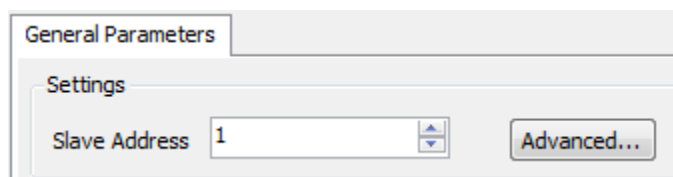


Figure 8-70. MODBUS RTU Slave for Symbolic Mapping Configuration Screen

Configuration	Description	Default	Possibilities
Slave Address	MODBUS Slave address	1	1 to 255

Table 8-21. 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 8-71 and Table 8-22. In addition, it is possible to select whether to keep the communication running when the CPU is stopped.

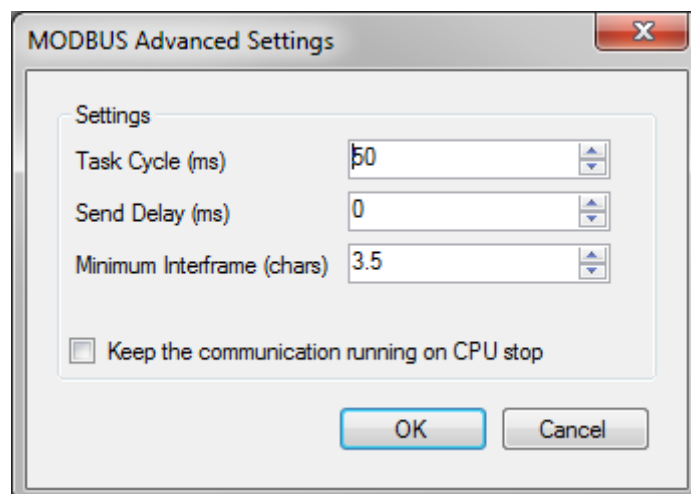


Figure 8-71. MODBUS RTU Slave for Symbolic Mapping Advanced Configurations Screen

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
Communication with the UCP stop	It keeps the communication running when the CPU is stopped.	Unmarked	Marked Unmarked

Table 8-22. MODBUS RTU Slave for Symbolic Mapping Advanced Configurations

Mapping Configuration

The mapping configuration of MODBUS RTU Slave for Symbolic Mapping, seen on Figure 8-72, follow the parameters described on Table 8-23:

Mappings

	Value Variable	Data Type	Data Start Address	Data Size	Data Range
*		▼			

Figure 8-72. 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 8-23. 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.

MODBUS Client Configuration

Client Settings

%Q Start Address of Diagnostics Area
21520

Size
20

Used range: %QB21520..%QB21539

Protocol
 RTU via TCP
 TCP

Devices

Instance Name	Destination IP	TCP Port

Add... Remove Edit...

Data Type	Slave Address	Read Data Start Address	Read Data Size	Read IEC Variable	Write Data Start Address	Write Data Size	Write IEC Variable

Add... Remove Edit...

Figure 8-73. 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 8-24. Diagnostic Initial Address

Adding a Device

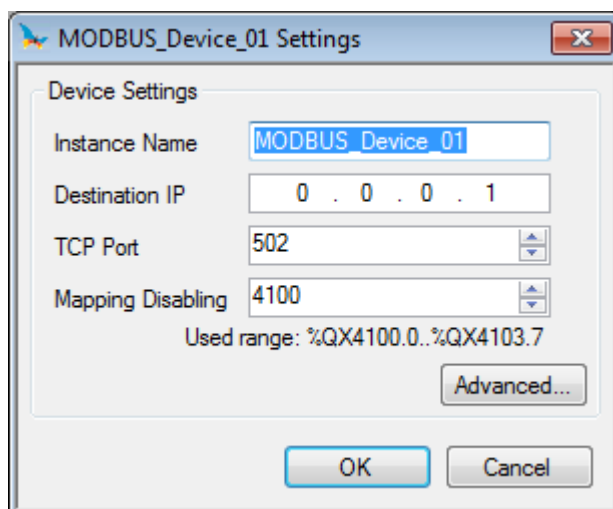


Figure 8-74. Adding a Device on MODBUS Ethernet Client for Direct Representation

Configuration	Description	Default	Possibilities
Instance Name	Instance name	MODBUS Device	Identifier, as IEC 61131-3
Destination IP	Server IP address	0.0.0.1	0.0.0.1 to 223.255.255.255
TCP Port	TCP port	502	2 to 65534
Mapping Disabling	Initial address used to disable the MODBUS relations configured in the Device Mappings field (%QX.X)	-	Any address of %Q area, limited according to CPU used

Table 8-25. MODBUS Ethernet Client for Direct Representation Configuration

Advanced Configurations

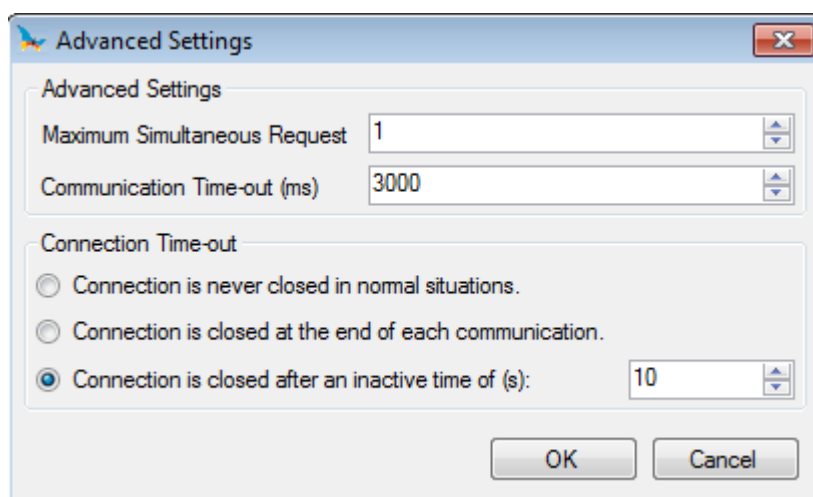


Figure 8-75. MODBUS Ethernet Client for Direct Representation Advanced Configurations Screen

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 8-26. 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 8-76), are defined as:

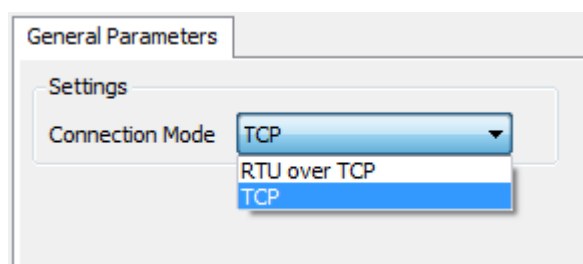


Figure 8-76. MODBUS Ethernet Client for Symbolic Mapping General Parameters Screen

Configuration	Description	Default	Possibilities
Connection Mode	Protocol selection	TCP	RTU over TCP TCP

Table 8-27. MODBUS Ethernet Client for Symbolic Mapping General Configuration

General Parameters

The configuration of the slave devices, seen on Figure 8-77, follow the parameters above:

Figure 8-77. 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 223.255.255.255
TCP Port	TCP Port	502	2 a 65534
Slave Address	MODBUS Slave address	1	0 a 255

Table 8-28. 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	Connection is closed after a time-out. Connection is closed at the end of each communication Connection is closed after a time of inactivity (s): 10 to 3600.
Time of Inactivity	Time of inactivity	10	3600

Table 8-29. MODBUS Ethernet Client for Symbolic Mapping Advanced Configurations

Mapping Configuration

The MODBUS Ethernet Client for Symbolic Mapping configuration, seen on Figure 8-78, follow the parameters described on Table 8-30:

Mappings					
Requests					
General Parameters					
	Value Variable	Data Type	Data Start Address	Data Size	Data Range
*					

Figure 8-78. 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 – Write (1 bit) Coil – Read (1 bit) Holding Register - Write (16 bits) Holding Register – Read (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 8-30. Mappings on MODBUS Ethernet Client for Symbolic Mapping

Requests Configuration

The MODBUS Ethernet Client for Symbolic Mapping request configuration, seen on Figure 8-79, follow the parameters described on Table 8-31:

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 8-79 MODBUS Ethernet Client for Symbolic Mapping Requests Screen

Configuration	Description	Default	Possibilities
Function Code	MODBUS function type	-	01 – Read Coils 02 – Read Input Status 03 – Read Holding Registers 04 – Read Input Registers 05 – Write Single Coil 06 – Write Single Register 15 – Write Multiple Coils 16 – Write Multiple Registers 22 – Mask Write Register 23 – Read/Write Multiple Registers
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 8-31 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.

MODBUS Server Configuration

Server Settings

%Q Start Address of Diagnostics Area: 21540 TCP Port: 502 Protocol: RTU via TCP TCP

Size: 20 Mapping Disabling: 4096 Advanced...

Used range: %QB21540..%QB21559 Used range: %QX4096.0..%QX4099.7

Server 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 8-80. 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 8-32. MODBUS Ethernet Server for Direct Representation Configuration

Advanced Configurations

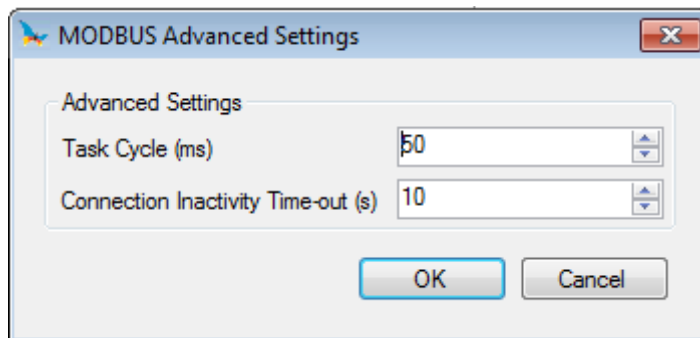


Figure 8-81. MODBUS Ethernet Server for Direct Representation Advanced Configurations Screen

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 8-33. MODBUS Ethernet Server for Direct Representation Advanced Configurations

Adding a MODBUS Relation

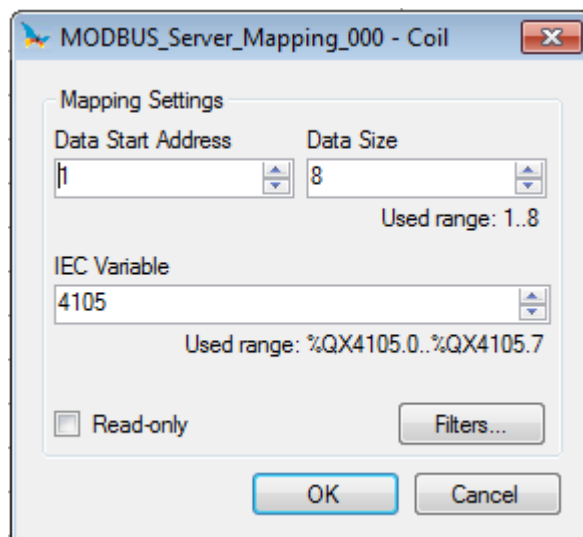


Figure 8-82. Adding a MODBUS Ethernet Server for Direct Representation Relation

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	0	1 to 65536

	address		
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 8-34. MODBUS Ethernet Server for Direct Representation Relation Configuration

Filters

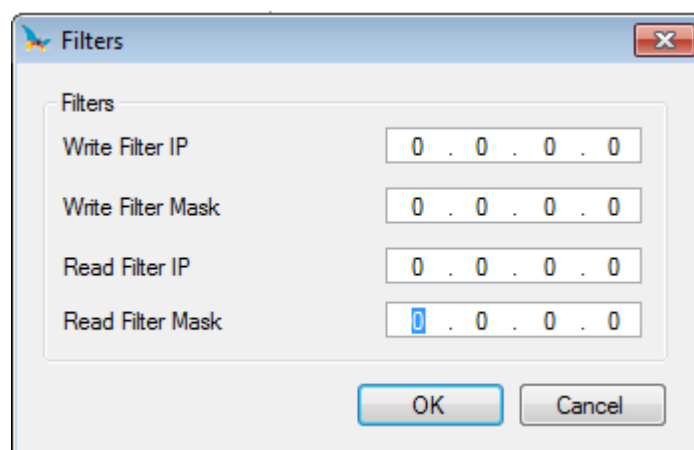


Figure 8-83. Filters

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	0.0.0.0 to 255.255.255.255
Write Filter Mask	Specifies a subnet mask and the parameter Write Filter IP	0.0.0.0	0.0.0.0 to 255.255.255.255
Read Filter IP	Specifies a IP interval with reading access to the declared variables in the MODBUS relation	0.0.0.0	0.0.0.0 to 255.255.255.255
Read Filter Mask	Specifies a subnet mask and the parameter Read Filter IP	0.0.0.0	0.0.0.0 to 255.255.255.255

Table 8-35. 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 8-84), are defined as:

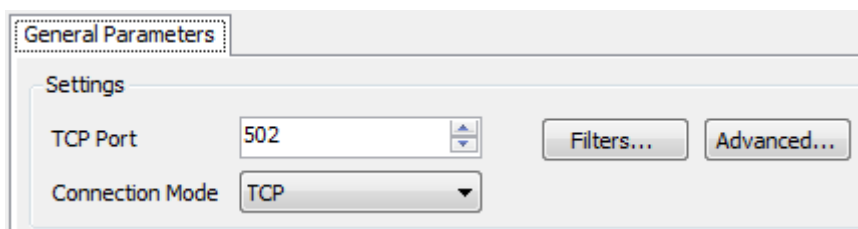


Figure 8-84 MODBUS Ethernet Server for Symbolic Mapping General Parameters Screen

Configuration	Description	Default	Possibilities
TCP Port	TCP Port	502	2 to 65534
Connection Mode	Protocol selection	TCP	RTU via TCP TCP

Table 8-36 MODBUS Ethernet Server for Symbolic Mapping General Configuration

The configurations in the *Filters...* button, described on Table 8-35, 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. In addition, it is possible to select whether to keep the communication running when the CPU is stopped.

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
Communication with the UCP stop	It keeps the communication running when the CPU is stopped.	Unmarked	Marked Unmarked

Table 8-37. MODBUS Ethernet Server for Symbolic Mapping Advanced Configurations

Mapping Configuration

The mapping configuration on the MODBUS Ethernet Server for Symbolic Mapping, seen on Figure 8-85, follow the parameters described on Table 8-38:

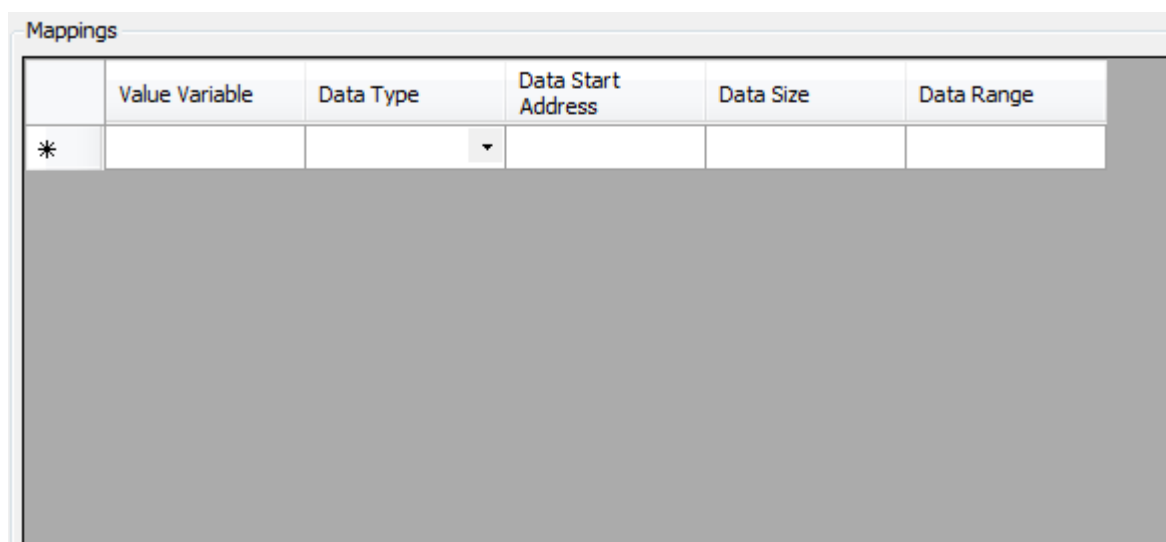


Figure 8-85. MODBUS Ethernet Server for Symbolic Mapping Screen

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 8-38 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.

The screenshot displays the 'General Parameters' tab of the CPU Editor. It is organized into several sections:

- Diagnostics Area:** %Q Start Address is 21520, Size is 520. Used range: %QB21520..%QB22039.
- Retain Area:** %Q Start Address is 4096, Size is 8192. Used range: %QB4096..%QB12287.
- Persistent Area:** %Q Start Address is 12288, Size is 8192. Used range: %QB12288..%QB20479.
- CPU Parameters:** Start User Application after a Watchdog Reset is set to 'Enabled'. Hot Swap Mode is set to 'Enabled, without startup consistency'.
- TCP/IP Parameters:** Initial Time-out (x100 ms) is 6, and ACK Delay (x10 ms) is 19.
- Project Parameters:** A checkbox for 'Consist retain and persistent area in %Q.' is unchecked. A 'Memory Card...' button is present.

Figure 8-86. 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.

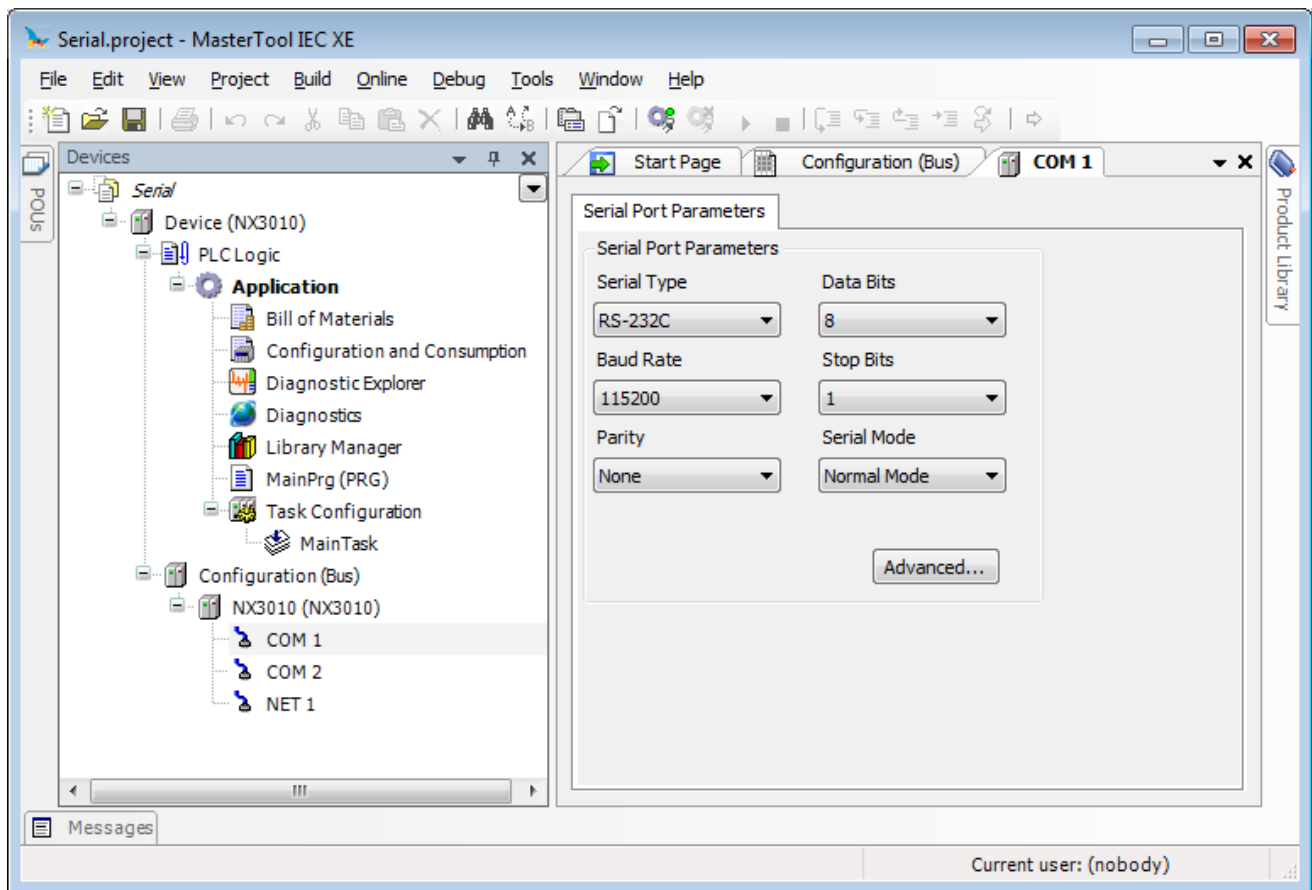


Figure 8-87. Serial Interfaces

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.

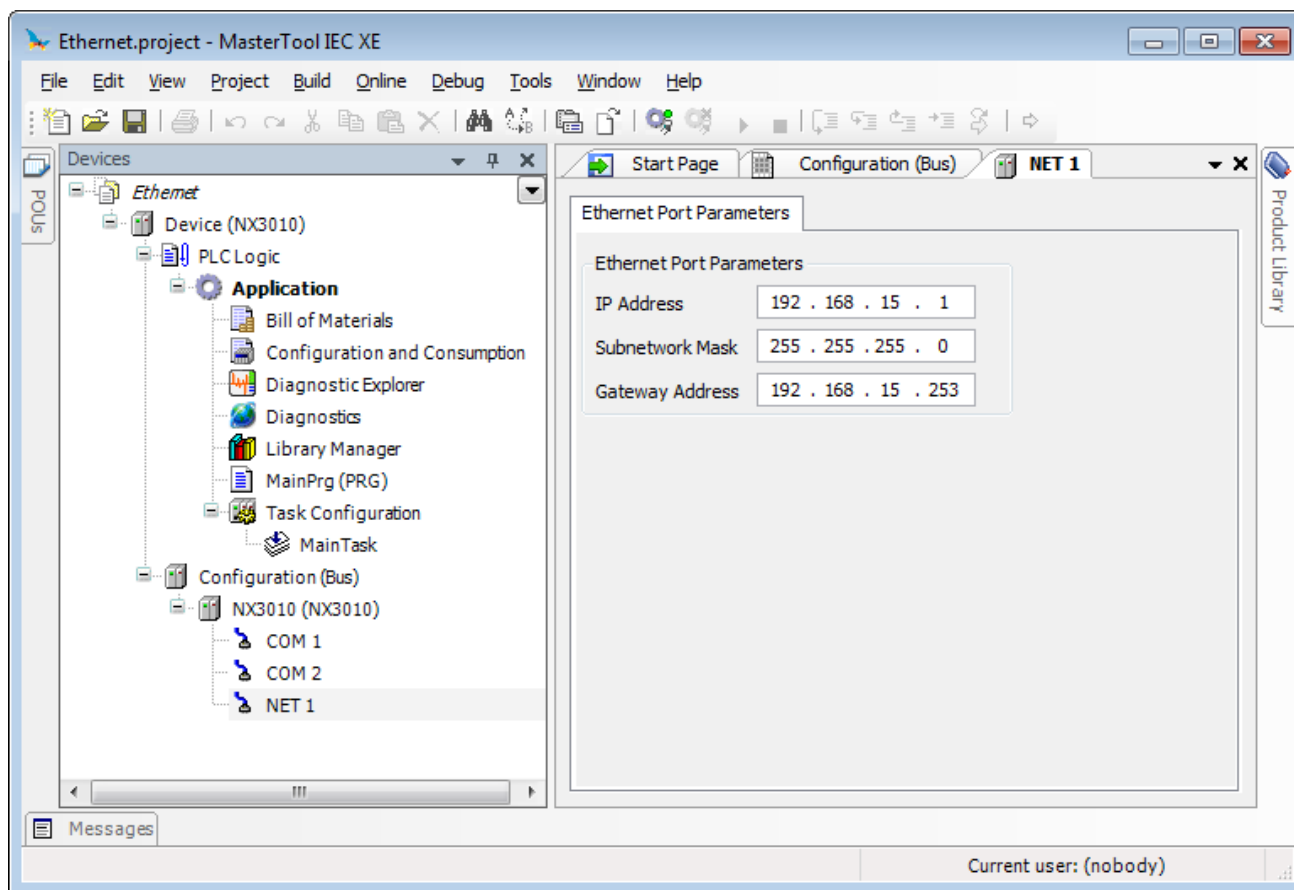


Figure 8-88. Ethernet Interface

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 8-89).

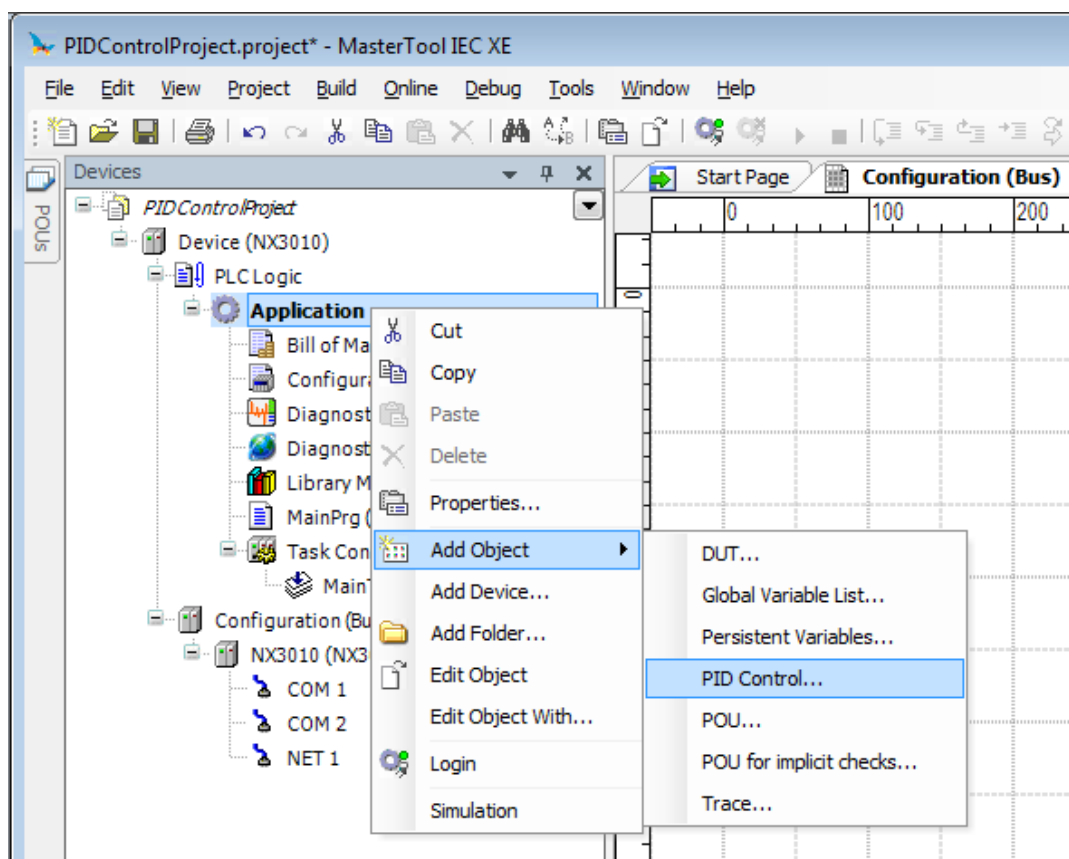


Figure 8-89. Steps to inserting the PID Control object in an application

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 8-90 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.

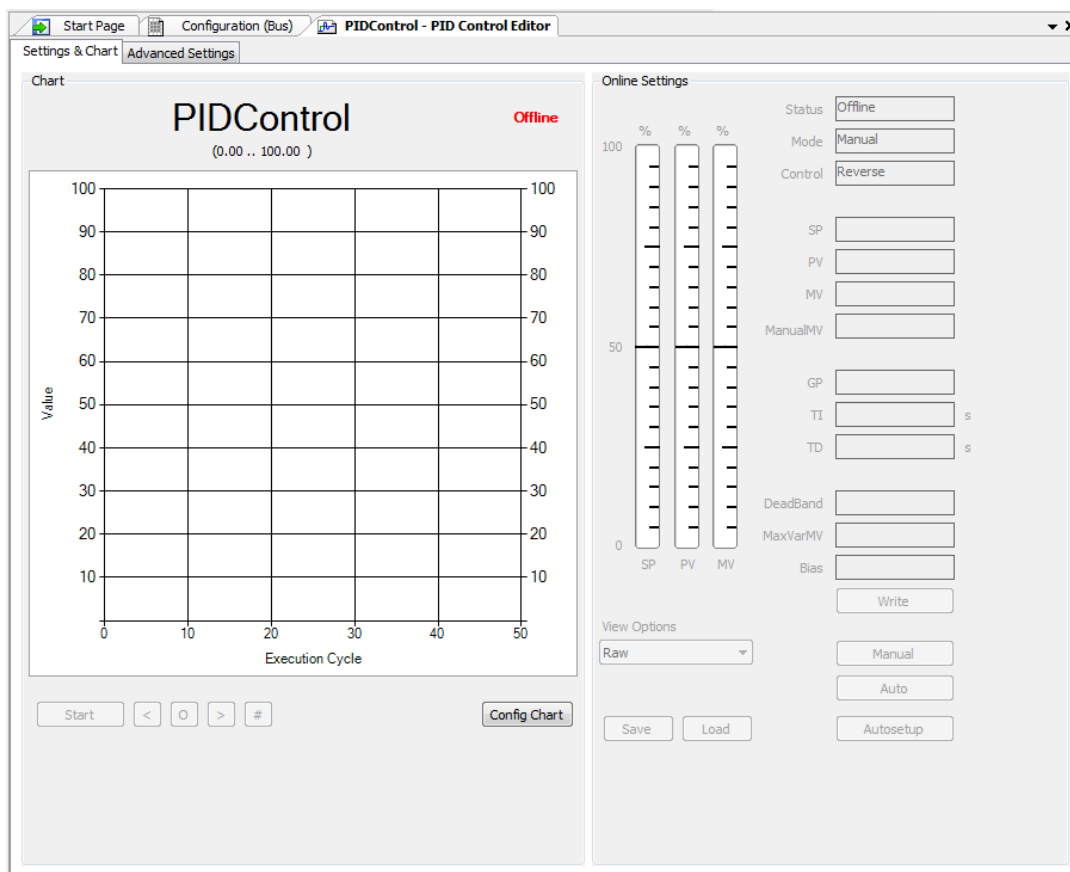


Figure 8-90. Graphical environment of the PID Control object

The graphical environment illustrated by Figure 8-90 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 8-91.

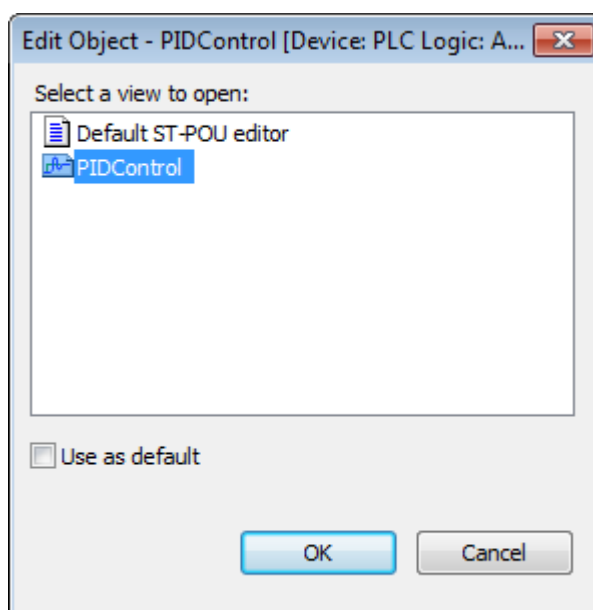


Figure 8-91. Access to the graphical environment of PID Control object

Tab Settings & Chart

The Figure 8-90 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 8-92 presents this group.

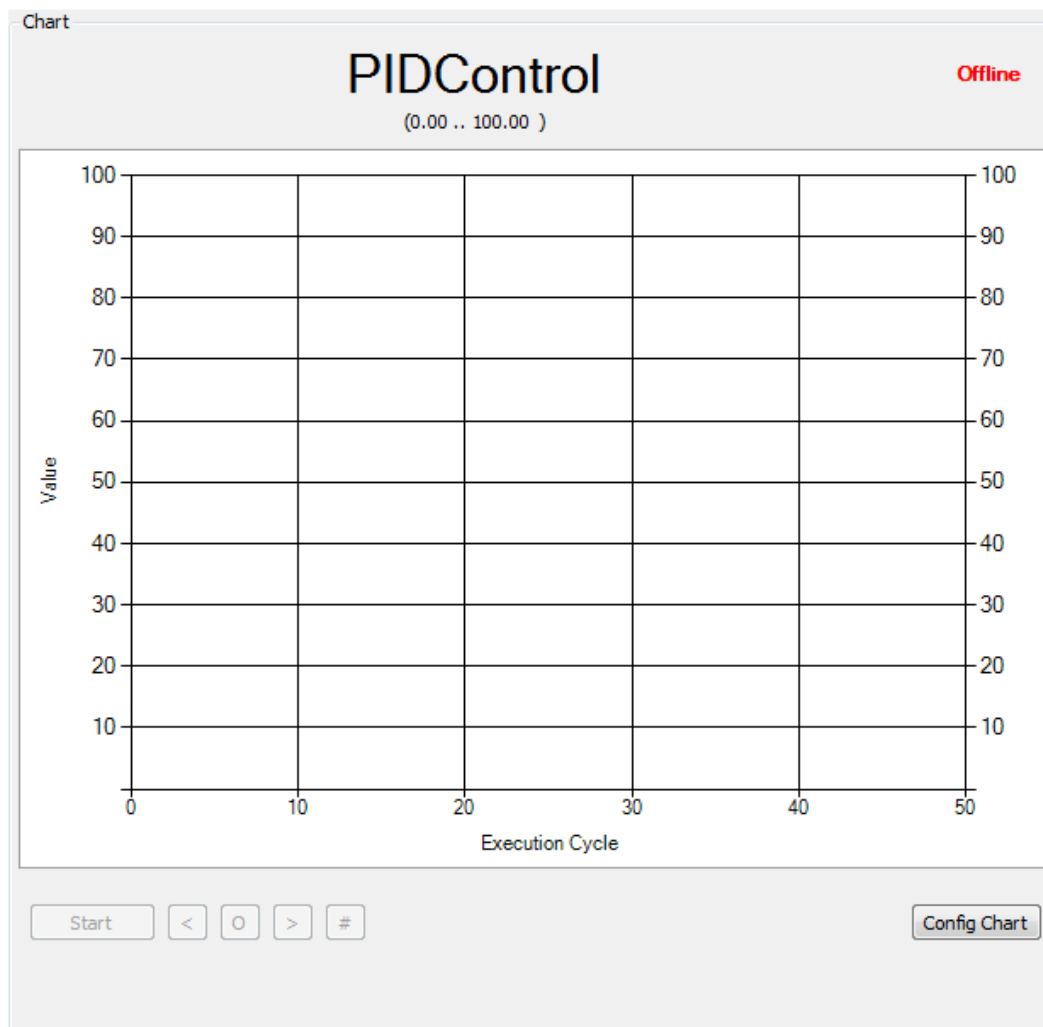


Figure 8-92. Group Graphic

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 8-93 shows these buttons.



Figure 8-93. 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 8-94.

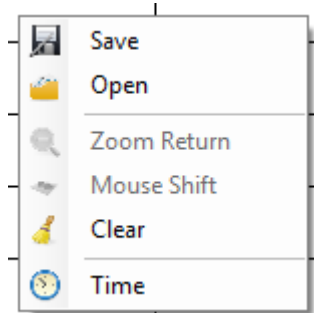


Figure 8-94. The Context Menu of the trend chart

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 8-95 shows this window.

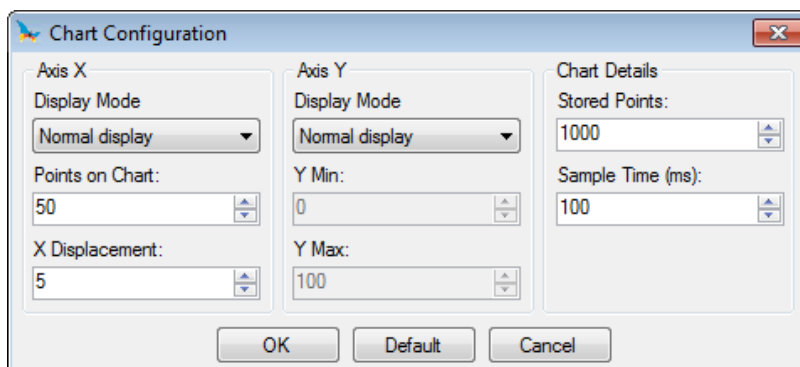


Figure 8-95. Chart Configuration Window

It can be observed that this window is formed by the *X Axis*, *Y Axis* and *Chart Details* groups.

In the group *X Axis* is possible to configure how the axis “x” will be displayed, with following options:

- *Normal display*: only the last k monitoring points are displayed in the chart, where k is configured in the field *Points on Chart*.
- *AutoFit*: all monitoring points are presented in the trend chart.

The *X Displacement* field specifies the step offset of the < and > buttons, located just below the trend chart.

In the group *Y Axis* is possible to configure how the axis “y” will be displayed, with following options:

- *Normal display*: the "y" axis will be displayed in the range of 0% to 100% (fields *Y Min* and *Y Max* remain disabled).
- *AutoFit*: the "y" axis will automatically adjust to display the data collected by monitoring (fields *Y Min* and *Y Max* remain disabled).
- *Personalized*: the range of values displayed by the "y" axis can be customized using the *Y Min* and *Y Max* fields.

In the group *Chart Details* is possible to configure the details of how to the chart will be displayed and stored, with following options:

- *Stored Points*: Sets the maximum buffer storage points on the graph for each process variable, after that limit points are discarded.
- *Sample Time (ms)*: Sets the time that each point on the graph is updated for each process variable.

The changes will only be confirmed by pressing the *OK* button. If you need to return to the default setting, the *Default* button must be pressed.

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 8-96 displays details of the *Online Settings* group.

The screenshot shows the 'Online Settings' window for a PLC. On the left, three vertical bar graphs represent the current values for Setpoint (SP), Process Variable (PV), and Manipulated Variable (MV) in percentage. The y-axis for these graphs ranges from 0 to 100, with a major tick at 50. All three bars are currently at 0%. To the right of the graphs, a list of parameters is shown, each with a text input field. The parameters and their values are: Status (Offline), Mode (Automatic), Control (Reverse), SP (0.00), PV (0.00), MV (0.00), ManualMV (0.00), GP (0.900), TI (10.000 s), TD (0.275 s), DeadBand (0.00), MaxVarMV (0.00), and Bias (0.00). Below the input fields are several buttons: 'Write', 'Manual', 'Auto', 'Autosetup', 'Save', and 'Load'. At the bottom left, there is a 'View Options' dropdown menu currently set to 'Raw'.

Figure 8-96. Online Settings group

In the upper left corner of the Figure 8-96, 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 8-97) 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.

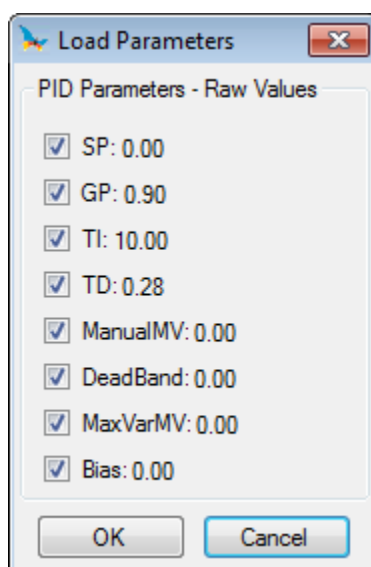


Figure 8-97. Parameters Selection Window

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.

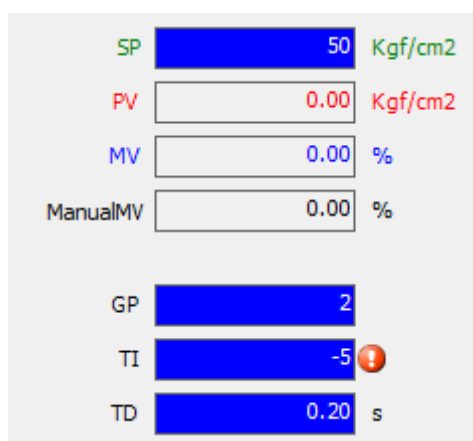


Figure 8-98. Parameters Editing Example

The Figure 8-98 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 8-99 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 8-99. 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 8-100.

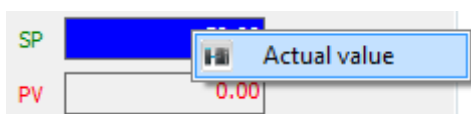


Figure 8-100. Restoring the Value of a Field

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 8-101 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): 100

Control: Reverse

Enable proportional action

Enable integral action

Enable derivative action

Enabled derivative action on PV

Project Settings

Autosetup Restrictions

Automatic Task Association

Figure 8-101. Advanced Settings tab

Observing the Figure 8-101 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 8-102 shows the group.

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	

Figure 8-102. 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.

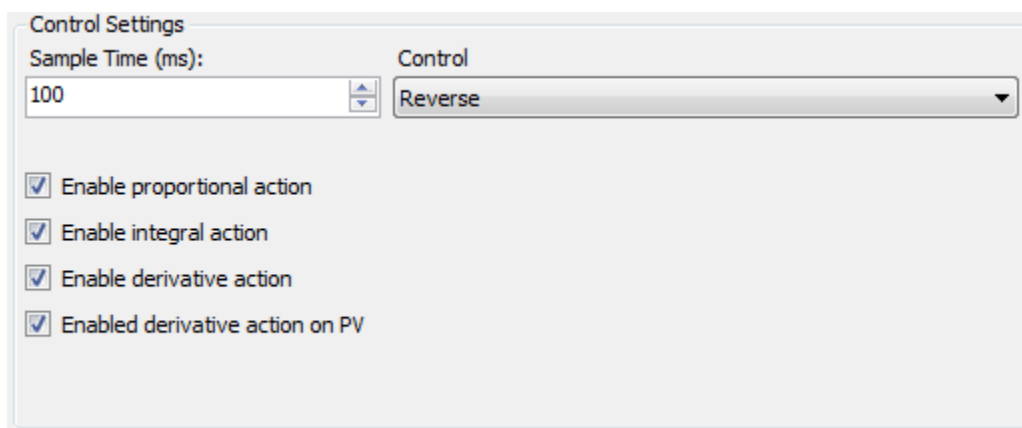


Figure 8-103. Group: Control Settings

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.

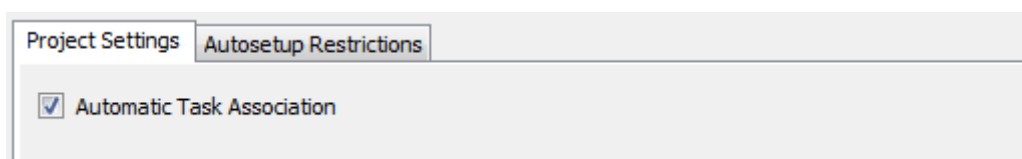


Figure 8-104. Tab: Project Settings

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.

Parameter	Value	Unit
Minimum MV	15	%
Maximum MV	85	%
Minimum PV	10	%
Maximum PV	90	%

Figure 8-105. 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 8-106 displays the window *Autosetup*.

Parameter	Value	Unit
Tf	1	s
Step	10	%
Method	PureStat	
Automatic Stop	<input type="checkbox"/>	
GP	0.90	
TI	10.00	s
TD	0.28	s
Gs		
Tde		s
Tc		s

Figure 8-106. 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 $(MV + MV * Step)$.

In the field *Method* the user can select between two methods to perform the identification of process parameters. The first of them *PureStat*, uses only statistical information, while the second *PolyStat* uses statistical information in conjunction with polynomial approximation. The statistical information is used to try to soften the presence of noise in the process

The *Automatic Stop* option enables the automatic termination of the experiment required to obtain the approximate parameters of a first-order model to be used by the method of synthesis. To determine the end the algorithm monitors the signal PV waiting for its stability. To wait for the stability the algorithm monitors the signal to each sampling. If the shifting from one sample to another is less than 2% of the total variation the algorithm waits 800 samples within the range to consider the signal stable. If it is perceived by graphical representation that the signal does not stabilize the experiment can be toggled to manual mode.

The fields *GP*, *TI* and *TD* of the *PID parameters* tab, express, initially, the current values configured for the controller parameters.

The fields *Gs*, *Tc* and *Tde* of the *Process* tab, express values of the static gain of the process, the time constant of the process and the deadtime of the process, respectively.

Before you start the experiment of identification of process parameters and tuning of the parameters of the controller, it is important that the process is in steady state. Guaranteed this situation the experiment can be started by pressing the *Start* button.

Having been the experiment started, the trend chart begins to monitor the process to ensure that the operator keep track of what is occurring. In the experiment, initially, the PID controller is passed automatically to manual operating mode.

Then, after a certain period of time the step signal is applied. It is important to note that the signal level is not applied immediately. In this interval in which the MV remains unchanged is being collected some statistical information that will be used to minimize the possible presence of noise in the process.

After applied the step, the process will begin to respond to this stimulus until steady state again. The experiment must be kept running with the process in steady state for certain period of time. The procedure can be stopped by pressing the *Stop* button. One should remember that being the *Automatic Stop* option is enabled at some point, the experiment will be terminated automatically by the tuning procedure. However, the operator is still able to terminate the experiment when convenient, even before the automatic finalizing.

When the *Stop* button is pressed the experiment is terminated and the new suggested parameters to the controller are displayed in *GP*, *TI* and *TD*. In order to send the parameters to the PLC the *Write* button must be pressed.

After the auto tuning procedure, the controller will remain working in its manual mode. To return to the automatic mode, press the *Auto* button of *Online Settings* group of *Settings & Chart* tab.

Visualization

If you want to simulate, operate or monitor machines or plants, then the fully integrated Visualization is a ideal solution. A modular visualization concept offers a visualization client which can flexibly and with very little effort be used for the most different customized applications. The visualization editor offers ready-made complete visualization elements supported by a range of commands. Use these elements to create modern visualization masks with only a few clicks. The elements can be organized in libraries.

Overview

Tools e Editors

A Visualization bases on a modular visualization concept and provides following tools/editors:

- **Visualization Editor:** here it is possible to create a visualization/mask/panel in the IEC 61131-3 tool doing only a few clicks.
- **Visualization Elements:** ready-made elements are available for use in a wide range.
- **Visualization Profile:** allows you to define a profile of visualization libraries. Each visualization project containing at least one visualization object must base on one.
- **Visualization Libraries:** here, elements can be summarized to make them suitable for use in other projects.
- **Project Settings:** settings for visualization projects.
- **Visualization Manager with Clients:** here, the visualization clients are managed and configured.

A Visualization with MasterTool enables you:

- to nest visualizations by references and/or toggle between different visualizations: **Frame;**
- to use interfaces for parameter transfer to instantiate complex visualizations: **Basics of Interface Editor;**
- to support multi-language by an integrated text list editor: **Text and Language in Visualization;**
- to install an user management system: **User Management.**

General Mechanism

The remote visualization clients are nothing else but interpreters of draw commands. Each client will get the same instructions, so that the resulting visualizations will all be the same.

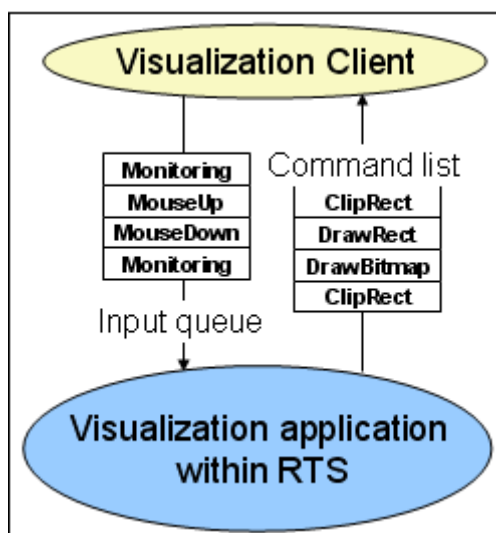


Figure 8-107. General Mechanism

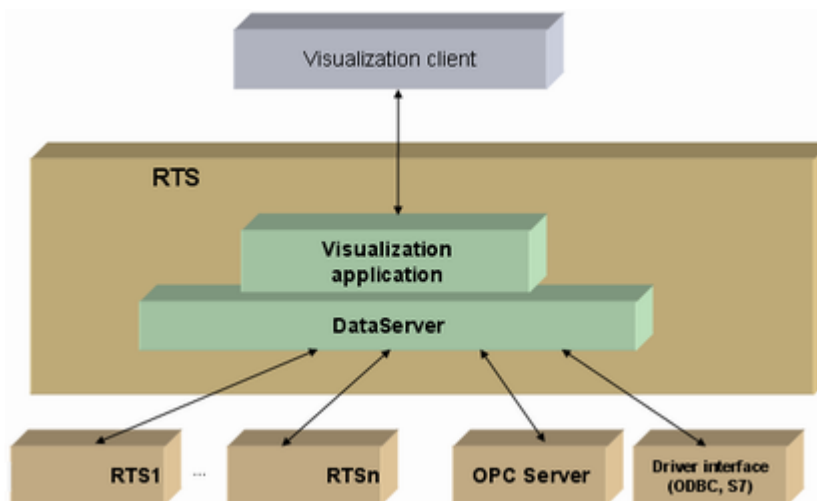


Figure 8-108. Data Server for Providing Visualization Data

A visualization application are created in IEC code in the visualization editor of the programming system. In case the visualization is running on the device, visualization code will be generated and loaded there.

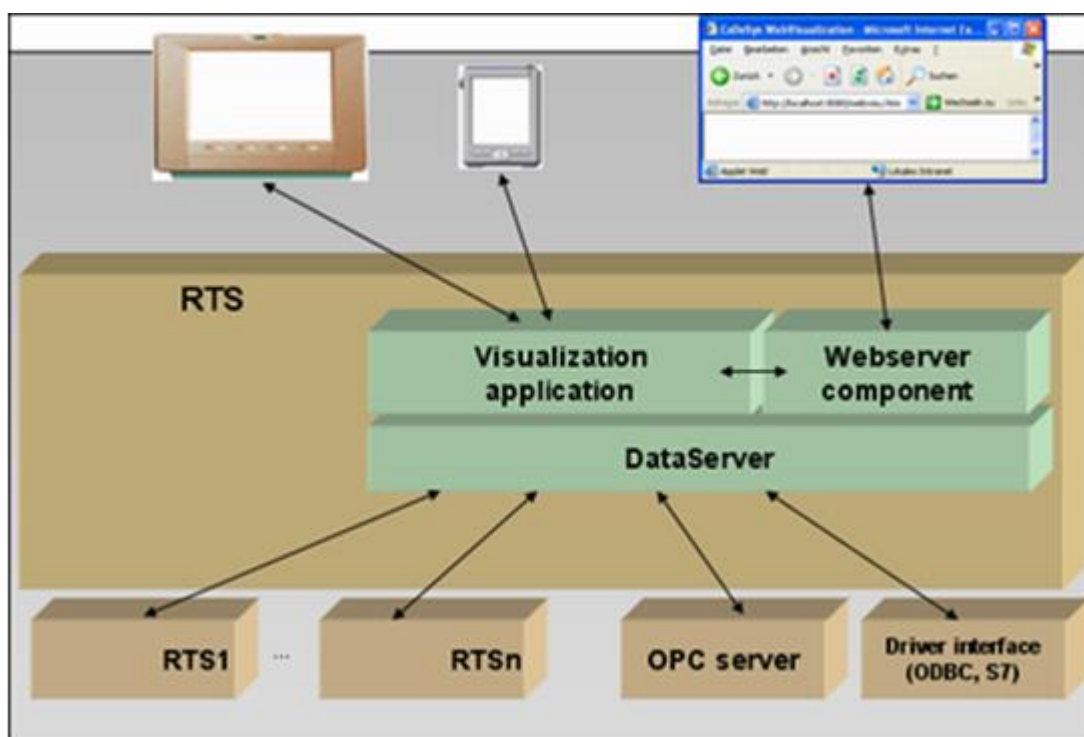


Figure 8-109. Visualization Client WebVisu

WebVisu is for using a web server to connect to the related application.

External data sources (remote) can be used in visualizations. For this purpose, a data server must be available for local application dealing with respective data sources.

ATTENTION:

The WebVisu and Application Download are two processes which spend a lot of memory, when the two run together, it's normal that the free memory decreases significantly. In some cases will have a message in log: *Free memory reached critical condition*.

Orientate do make a *Reset Origin* or *boot without application* (button pressed) before to do a download when the problem happens..

Create a Visualization

A visualization object can be inserted application-assigned in the device tree below an application or in the POU-pool by using the command *Project > Add Object ...* (also context menu). The window of a new visualization object is titled with *<visu object name>* and shows an empty area at first.

When the first visualization object is added in the project, the respective Visualization Libraries will be added automatically in the Library Manager. A visualization library always is set up as a special type of *placeholder library* which means that the exact version of the library to be used is not resolved until the library gets included in the project. Only then the currently active profile will define which version actually is needed. Notice that this type of libraries is different from the device-specific *placeholder libraries* where the placeholders get resolved by the device description.

The basic libraries which are included by default when adding a visualization object in a standard project are VisuElems, VisuElemMeter, VisuElemWinControls, VisuElemTrace, VisuInputs and further libraries additionally included of these basic libraries. Usually you do not have to include the visualization libraries manually or to use them explicitly in your applications.

A visualization can be created as an object in the global object pool, that is in the POUs window, or in the Devices window, directly assigned to an application. As soon as the first visualization is added below an application, a **Visualization Manager with Clients** will be added.

A Start Visualization, that is the visualization object which should be opened first after login on the PLC with an application, must be inserted in the Devices window below the respective application object.

Visualization, Dialog or Numpad/Keypad

Each particular visualization can have **Properties of a Visualization Object** like its designated use: *Visualization, Numpad/Keypad, Dialog* or the display size. Notice in this context that a visualization might be created and configured explicitly for being used as user input dialog (**Input Configuration**) in other visualizations. Implicitly also a standard numpad and a keypad mask are available for this purpose. The use of such keypads and dialogs can be defined in the property **Input Configuration** of a visualization element..

Note:

Getting online: The visualization works with help of an integrated runtime system. So during primary editing actions, you will get messages on starting and downloading.

Simulation Mode: The Web Visu is not available to in Simulation Mode.

Visualization Commands

The visual commands are provided with the *Visual Editor* plug-in for the category *Visual Commands*. They serve to edit a *Visualization Object* of the visualization editor. Most of them are part of the **Menu Visualization** and usually also of the context menu in the visualization editor. Follow the Visual commands that are not listed on the **Menu Visualization**:

Add Visual Element

This command (category *Visual Commands*) is available in the Visualization menu and in the context menu. It is used to insert an element into the current visualization and thus corresponds to inserting via drag-and-drop from the **Toolbox**.

The command opens a submenu where from a list of all currently available visualization elements one can be chosen. The selection matches that of the visualization **Toolbox**.

Frame Selection

This command (category *Visual Commands*) is used to configure the content of a **Frame**. A frame element is used to define a subarea of a visualization which includes one or several other visualizations. In online mode it can be toggled between the display of these particular visualizations. Basically the first visualization in the list of visualizations assigned to the frame is displayed. But by an input on another - appropriately configured (**Switch Framevisualization**) - visualization element the user might effect that one of the other assigned visualizations will be displayed in the frame. Thus switching between different displays within a visualization is possible.

The visualizations included in a **Frame** are references (instances of the original visualizations). Placeholders, defined in the original visualization objects can be replaced by locally applicable values.

The selection of the visualizations in a frame is done in the dialog **Input Configuration**, which is opened by this command.

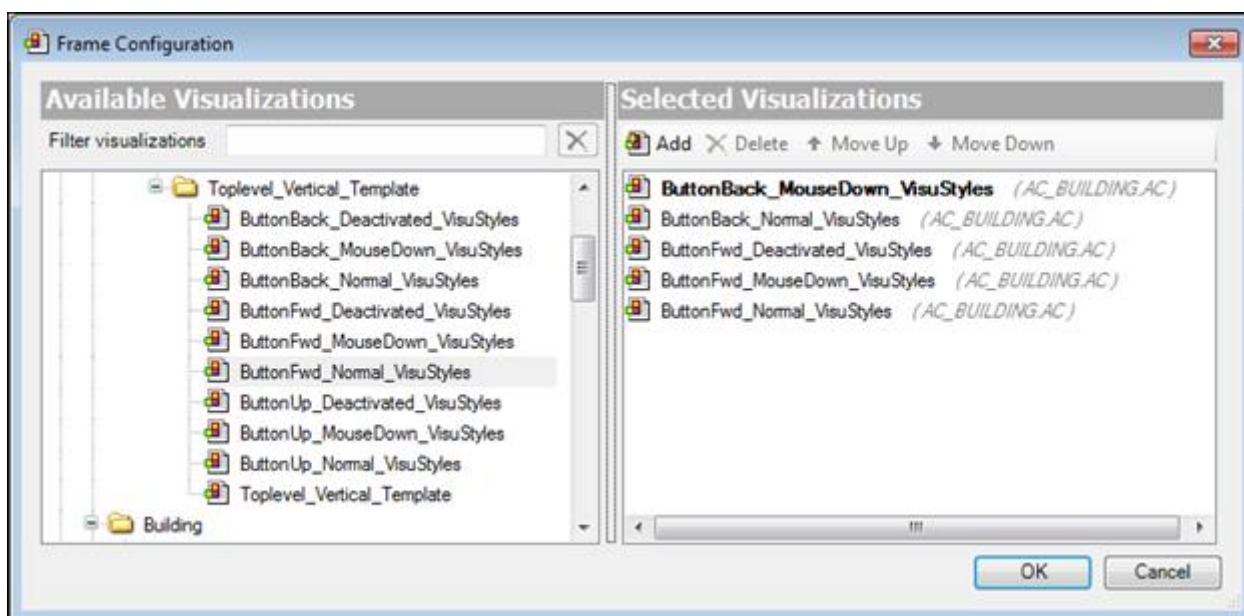


Figure 8-110. Frame Configuration Dialog

On the left side the *Available Visualizations* of the project are listed. Select those which should be included in the frame and bring them to the *Selected Visualizations* list on the right side by a double click or the *Add* button. Selected visualizations can be removed by a double click or the *Delete* button. Multi selection of elements is also possible. The order within the list can be changed by the *Move Up* and *Move Down* command.

It is recommended to assign only those visualizations to a frame, which are managed in the global pool. Otherwise problems might occur, if at a later time any device or application objects will be renamed and due to this the path of the assigned visualizations will not be valid any longer.

The order of the selected visualizations from top to down determines the automatically generated implicit index numbers for the visualizations. The uppermost gets index 0, the following ones 1, 2 etc.. The index numbers are required for the configuration of the switch function Switch Framevisualization of another element. Initially the visualization with index 0 will be displayed.

Example:

Via a menu bar you want to determine which of several visualizations currently should be displayed in online mode.

1. Add a frame element
2. Assign this frame via *Frame selection* in Dialog *Frame configuration* the visualizations to be toggled.
3. Add control elements for each visualization which allows to switch to this visualization. For example add a menu bar with control elements.
4. Each control elements is configured by setting its **Input Configuration**: Add in property OnMouseClicked the after action **Switch Framevisualization** and assign there the particular visualization to the particular frame.

Select All

Symbol: 

This command (category *Visual Commands*) is used to select, at one time, all elements of the current Visualization.

Select None

Symbol: 

This command (category *Visual Commands*) is used to cancel, at once, all selections in the current Visualization.

Visualization Editor

The visualization editor is available via the *Visualization Editor* plug-in.

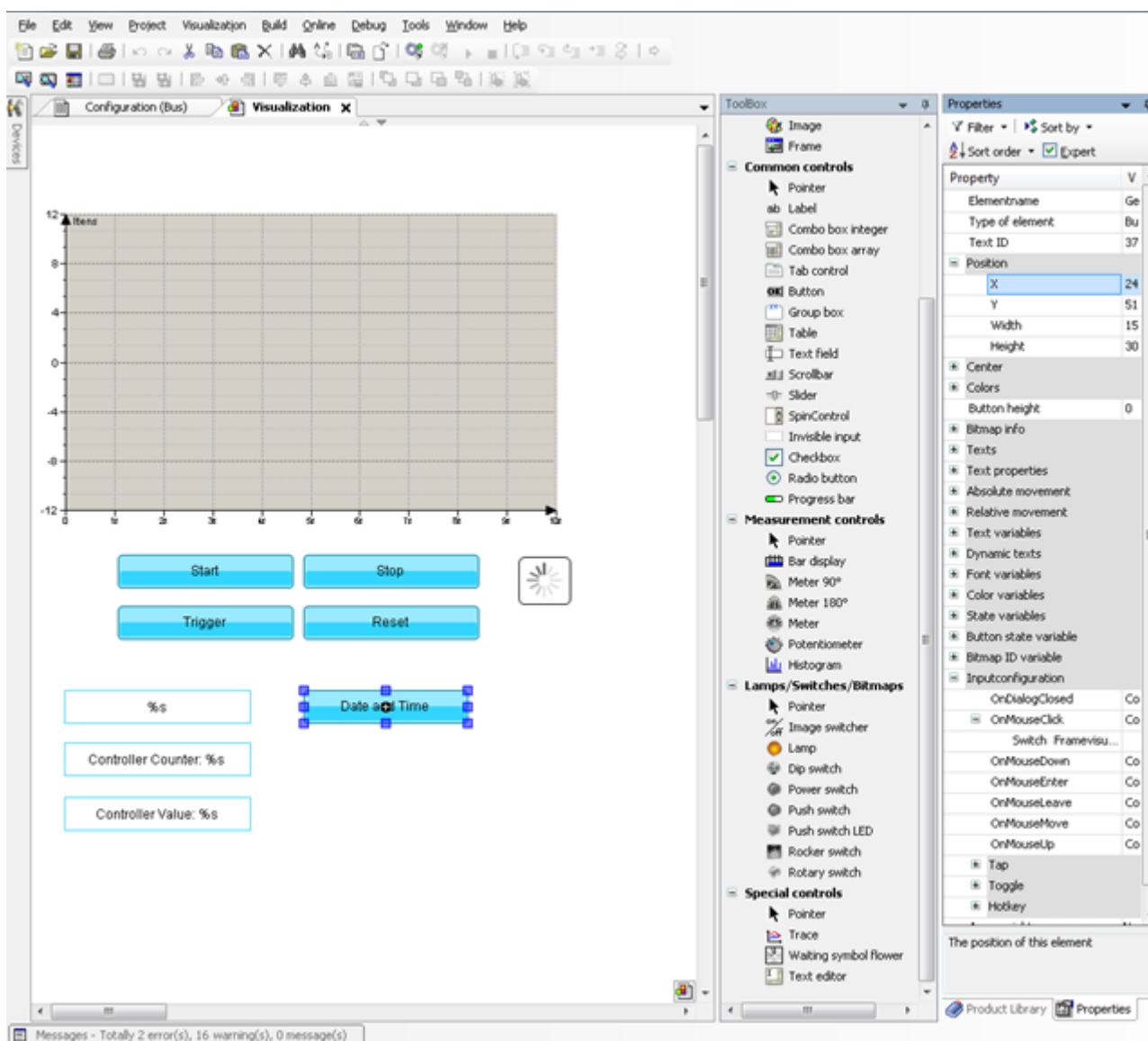



Figure 8-111. Editor de Visualization

A visualization can be created with help of following editors and additional tools:

- The editor window is titled  <visualization name>[<object path>] and displays the painted/edited/programmed visualization.
- The *Toolbox* which contains all currently available visualization elements.
- The *Properties* window containing the properties of the visualization element which is currently selected in the editor window.
- The commands provided by the *Visualization menu* or by context menu in the editor.
- The *Interface Editor* for defining placeholders. If the visualization is intended to be inserted in another visualization.
- The *Hotkey Configuration* for assigning actions to keys or key combinations. Regard however that the device using the visualization must support the respective keys.
- The *Element List* providing an overview on all elements of the current visualization and allowing selection, deletion and changing element position from back to front and vice versa.

If the *Interface Editor*, *Hotkeys Configuration*, or *Element List* is activated via the respective command (per default in the *Visualization menu*), the Visualization Editor will get bipartite and provides the respective tabs in the upper part.

The visualization elements can be animated by the direct use of the project variables, or in the form of expressions, that is combined with operators and constants. For example this allows to scale the variables for the usage in the visualization.

Within a visualization arbitrary expressions, even function calls are valid.

Tool Box

The *ToolBox*, which is used in combination with the Visualization Editor, provides visualization elements for insertion in the editor window.

The elements are provided via **Visualization Libraries** and the current selection depends on the currently active **Visualization Profile** and in case of device-associated visualizations by the device description.

The **Toolbox** can be opened via the **View Menu**. It consists of the following categories:

- **Basic**
- **Common Controls**
- **Measurement Controls**
- **Lamps, Switches, Bitmaps**
- **Special Controls**

There the associated visualization elements are listed with names and icons. Via drag-and-drop the elements can be inserted in the currently opened visualization editor window. A plus-sign at the cursor during dragging indicates that the element will be brought into the editor window. After having been dropped, the element will be displayed in the visualization.

For inserting other visualizations in a frame element of the current visualization use **Frame Selection** command.

The standard libraries for visualization elements are: **Visualization Libraries**.

Element Properties

Symbol: 

The *Visualization Element* plug-in provides the command *Element Properties*, presented on a **View Menu**, which is used to open the editor *Properties* for a selected visualization element.

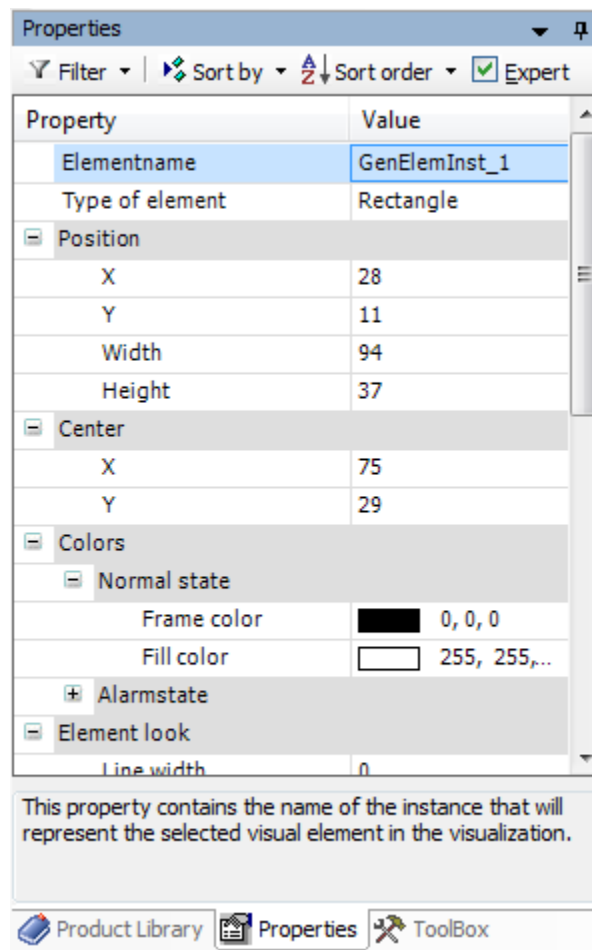


Figure 8-112. Element Properties Editor

The *Properties* editor list all properties of a selected element in a table and provide different user input possibilities for changing the *Value* of it. A description of the properties and its use is listed with the element description: **Visualization Elements**.

The menu bar at the top provides commands and options:

- *Filter*: For selection of the filter settings in order to adapt the properties list to your needs.
 - *Show all categories*: all properties
 - *Simple*: only the most basic properties Text, Color, and Input configuration are listed.
 - *Standard*: the mostly used properties (selection bases on the property set used in the programming system)
 - *Animation*: only properties used for animation
 - *Color*: only color referring properties are listed.
 - *Texts*: only texts referring properties
 - *Inputs*: only user input referring properties
- *Sort by*:
 - *Sort by type*: the complete list of properties is arranged in the original order of categories.
 - *Sort by name*: the complete list of properties is arranged alphabetically concerning the category names.
- *Sort order*:
 - *Sort ascending*: the complete list of properties is displayed in the original sequence of categories from down to up.

- *Sort descending*: the complete list of properties is displayed in the original sequence of categories from up to down.
- *Checkbox Expert*: with activation of the checkbox all properties are listed and Show all categories is set.


User Interface when Editing the Properties

Properties define the characteristic of a visualization element:

- Its appearance: changes on properties responsible for the appearance of the element are immediately visible in the visualization editor.
- Its arrangement in the visualization.
- Its animation: the visualization element can be animated by direct use of project variables.
- Actions to be executed after an user input on the element.

A property can be modified by editing the *Value* field. For this purpose depending on the property type, you can open an edit frame, a selection list, a dialog, or activate a checkbox. Such a widget will open



- by a double-click,
- by a single click in a selected field,
- or via [Space] bar when the field is already selected.

If a variable has to be assigned, use  for opening the *Input Assistant*. *Category Variables* lists all variables defined in the project up to know.

Interface Editor

This is used for defining an interface in a visualization which is intended to get referenced in a frame in another visualization.

Basics of Interface Editor

This part of the  Visualization Editor can be switched on or off via command  Interface Editor (by default in the *Visualization* menu). It will be provided in the upper part of the Visualization editor as a separate tab.

Declaring an Interface

Due to the fact that a visualization is handled as a function block, the Interface editor will be added as a usual declaration editor in the upper part of the Visualization window. There you can define input and in/out parameters also called frame parameters.

Declarations of parameters by:

- VAR_INPUT
- VAR_INPUT with **Attribute Parameterstringof**.
- VAR_IN_OUT: if a data structure is transferred, it has to be defined as VAR_IN_OUT. Without pragma, the values of the parameters will be copied when the visualization is opened. Inputs will be written on the copied data structure and with closing the visualization the values will be written back into the parameters.
- VAR_IN_OUT with **Attribute VAR_IN_OUT_AS_POINTER**: here, it is allowed to pass objects to visualizations as references. This is used for Diagnosis, Web and Target Visualizations. This can be useful if information has to be passed to a visualization without copying, or when information is updated from outside while the dialog is open.

ATTENTION:

If you add input variables with any of these parameters in the visualization interface editor, this visualization is expected to be used as a dialog or embedded into a frame. Then this interface is used to transfer data from the overlying visualization to the embedded visualization. A visualization with input variables can not be used as start visualization.

Declaring a Interface UserData

The visualization ViewData has the interface UserData of type IE_Data. The element GenElemInst_3 uses UserData and displays the variable value of UserData.iCounter like it is programmed in the properties view of *GenElemInst_3*.

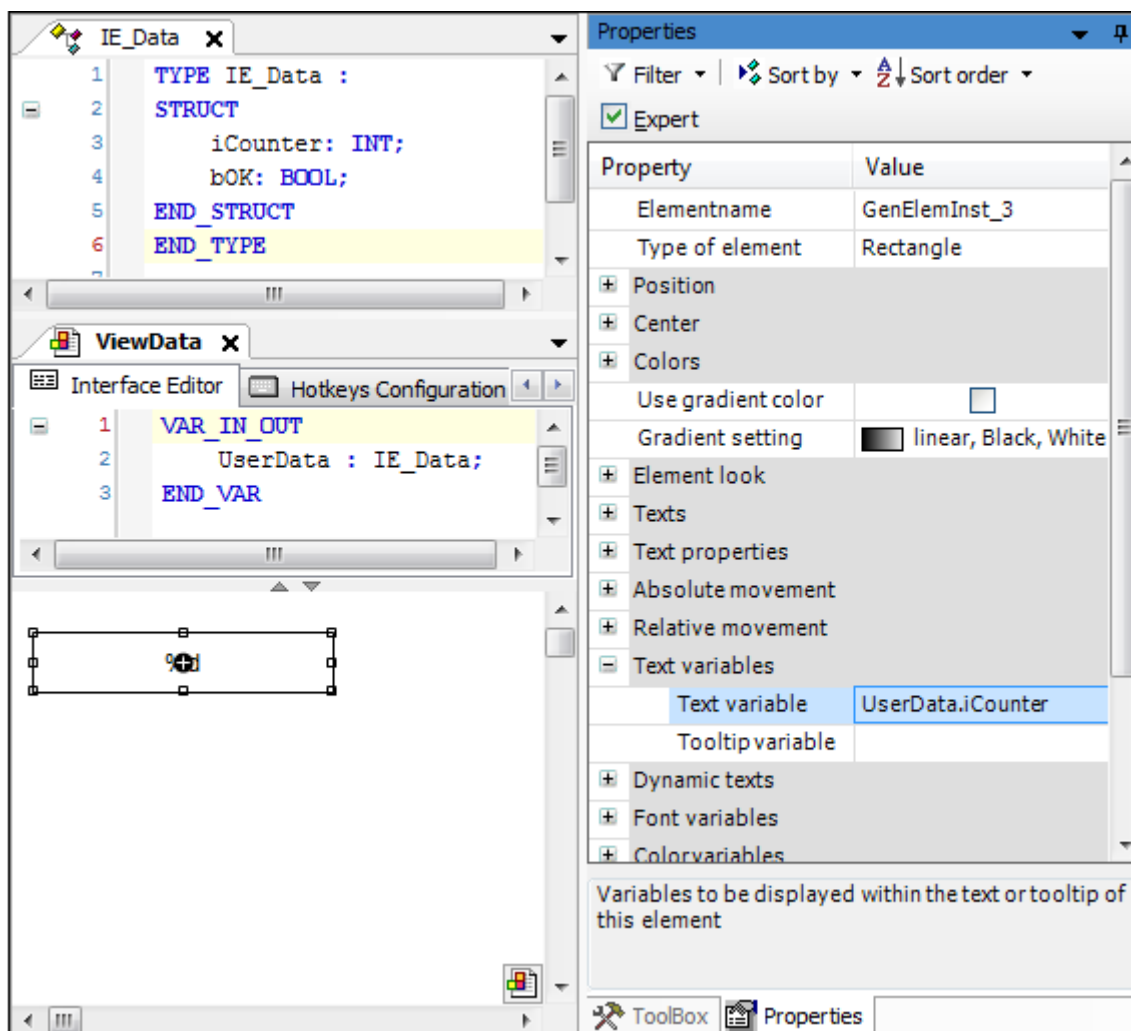


Figure 8-113. Declaring a Interface

Programming the Frame

1. Add a frame element in your main visualization.
2. Set the reference on a visualization with interface, which may be not only local but global or from a library.
3. Assign an IEC variable/expression to each frame parameter.

If a visualization with interface is referenced in an other visualization, then these frame parameters can be programmed by assigning locally used expressions in the

Properties view of the frame element that includes an visualization. The included (instantiated) visualizations and their variables are listed there and they can get assigned to any expressions which then can be used in the configuration of the respective instance..

Programming a Frame which References ViewData

The element **Frame** references the visualization ViewData. In the Properties view this assignment is listed with all frame parameters. With click in the value field of such a variable, an IEC variable of same type can be assigned. Here, Hans.HansData is assigned to UserData.

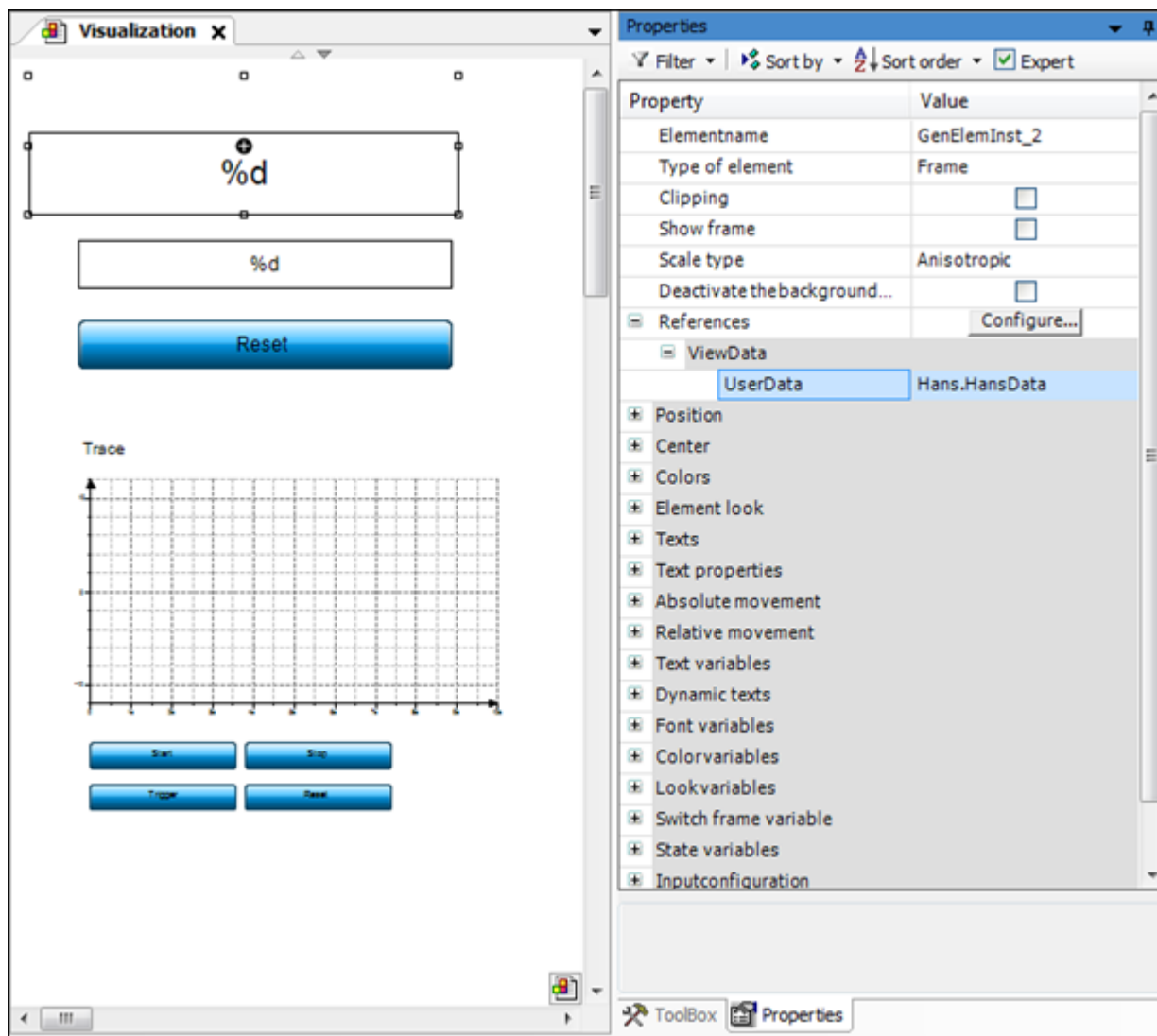


Figure 8-114. Frame References ViewData

Updating the Frame Parameters

If the interface of a visualization referenced in a Frame Element of another visualization gets modified, this dialog opens automatically when you try to save or compile the project or you try to open a concerned object. In this dialog you can (re-)configure the frame parameters.

In the case that the interface of a visualization coming from a library has changed, the parameters must be updated in your project as well.

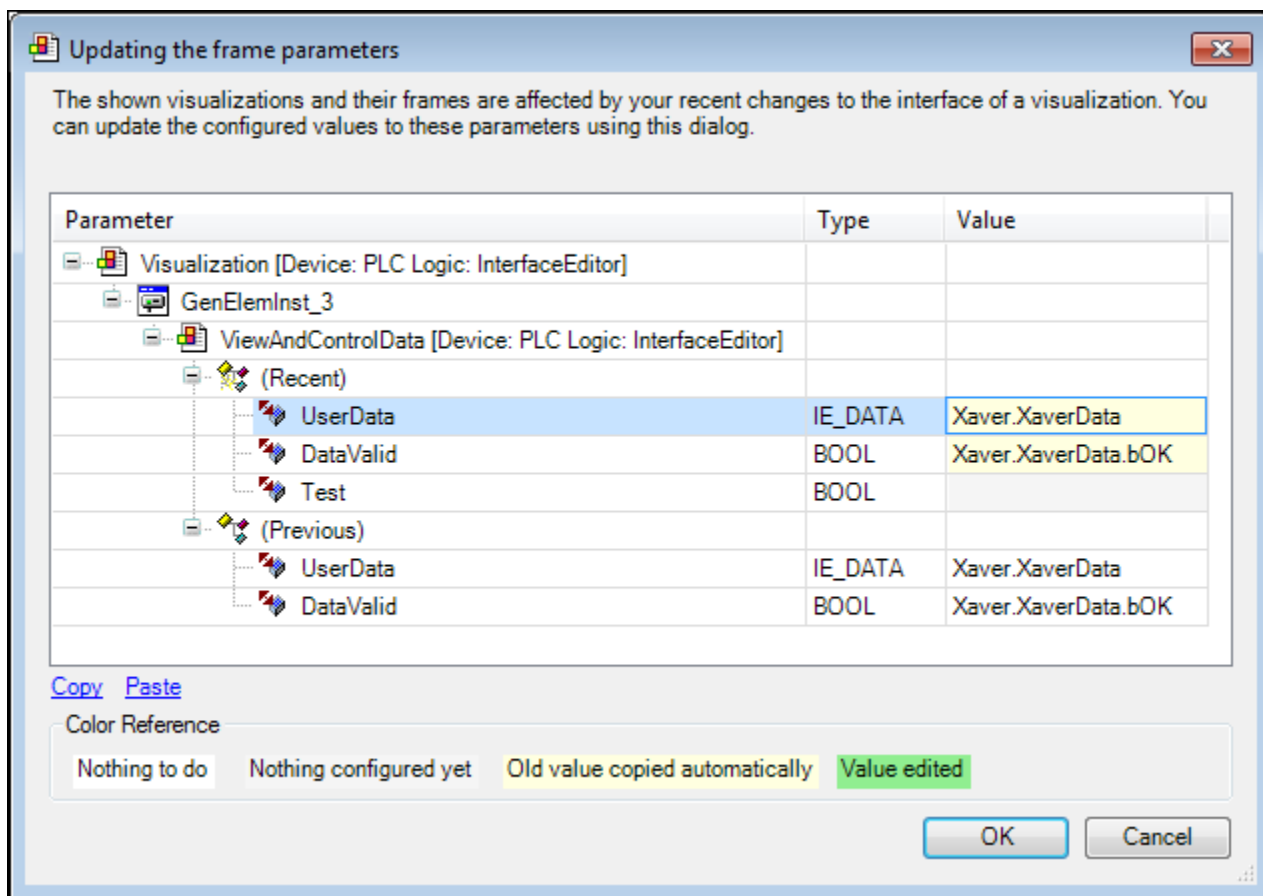


Figure 8-115. Updating the Frame Parameters

Parameter: in this column you see the changed parameters in tree view. On top there is the visualization with the changed frames. Below and indented, the name of the changed frame element is listed. Then, the visualization included by the frame is listed with its interface. Under the item *Recent* the current parameters are listed, under item *Previous* the former declaration is listed.

Type: in this column the type of the parameter is shown.

Value: the value field contains the variable which is assigned to the parameter specified in the first column.

Color reference:

- *Beige shading:* this parameter has been copied from the previous configuration automatically.
- *Grey shading:* no value has been assigned to this new parameter yet.
- *Green shading:* a value has already been assigned to this new parameter.
- *White shading:* nothing must be entered here.

Edit the values listed under Recent: click in the value field to select it. After a further mouse-click on it or after pressing [Space], an inline-editor opens. Or just begin to type. Assign another variable by typing its name or select one by help of the input assistant. An existing value-entry can be copied to another field. For that, select the field containing the entry to be copied and use button Copy, then select the field where the entry should be pasted to and use button Paste.

Confirm the settings in the update dialog by [OK]. The dialog closes and the new parameter assignments will be displayed in the *Properties* dialog, category *References* of the respective frame element.

If you want to make use of the instance name of an input variable passed in a referenced visualization, you will have to work with the **Attribute Parameterstringof** providing a corresponding string.

Adding Parameter DataValid

The interface of visualization `ViewAndControlData` is extended by variable `DataValid`.

Interface Editor

```
VAR_IN_OUT
```

```
  UserData: IE_Data;
```

```
  DataValid: BOOL;
```

```
END_VAR
```

Now, if you try to compile or save the project or open *Visualization*, the dialog for the configuration of the new parameters will appear:

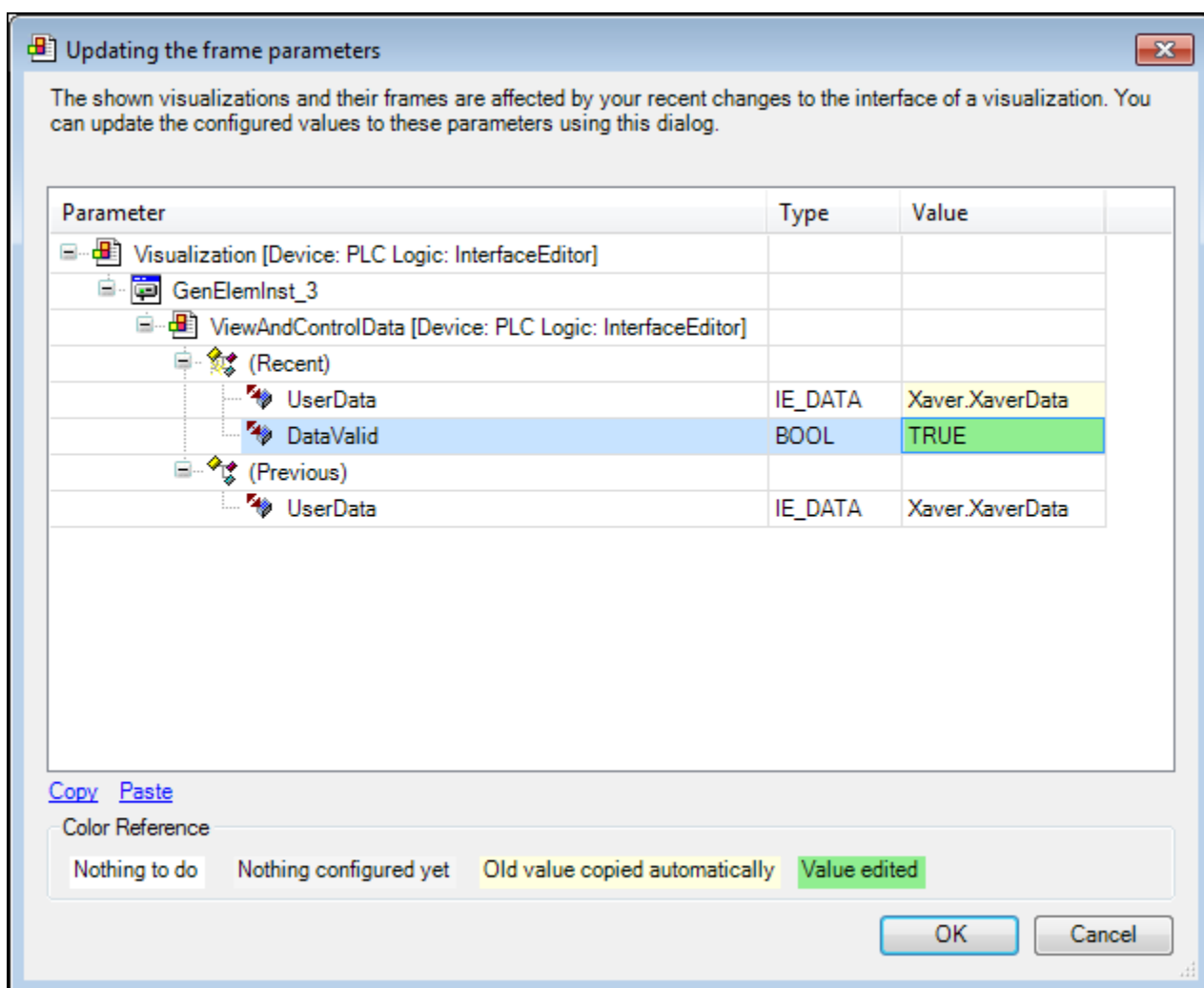


Figure 8-116. Updating the Frame Parameters

Compare the recent with the previous parameter. Enter `TRUE` in the value-field of `DataValid`. Close the dialog with `OK`. The new parameter assignments can be comprehended in the *Properties* view category *References* of `GenElemInst_3`.

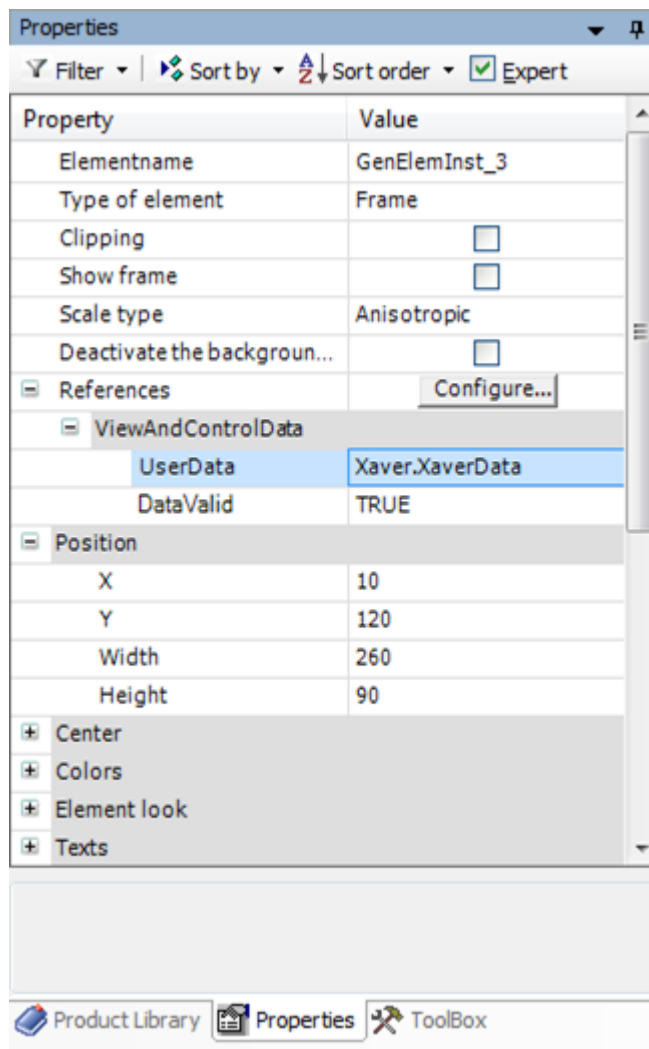


Figure 8-117. Properties View after Finished the Update of the Frame

Pragmas

Attribute VAR_IN_OUT_AS_POINTER

With help of this pragma, it is allowed to use a reference of an object as frame parameter. That's possible in a visualization used as dialog. Syntax:

```
{attribute 'VAR_IN_OUT_AS_POINTER' '}'
```

To use references, proceed as follows:

1. Declare your user data object (DUT).
2. Define a VAR_IN_OUT frame parameter in the interface editor of a dialog as a reference of an data object and give it the attribute 'VAR_IN_OUT_AS_POINTER'.
3. Program the user interface: use the dialog in an visualization or assign the dialog in the input configuration of an visualization element. Access to the referenced data is available.

Using of an interface with pragma VAR_IN_OUT_AS_POINTER:

```
FUNCTION_BLOCK ControlFB
VAR_INPUT
END_VAR
VAR_OUTPUT
```

```

END_VAR
VAR
    bOk : BOOL := TRUE;
    nCounter : INT;
    nValue : INT;
END_VAR

nCounter := nCounter + 1;

```

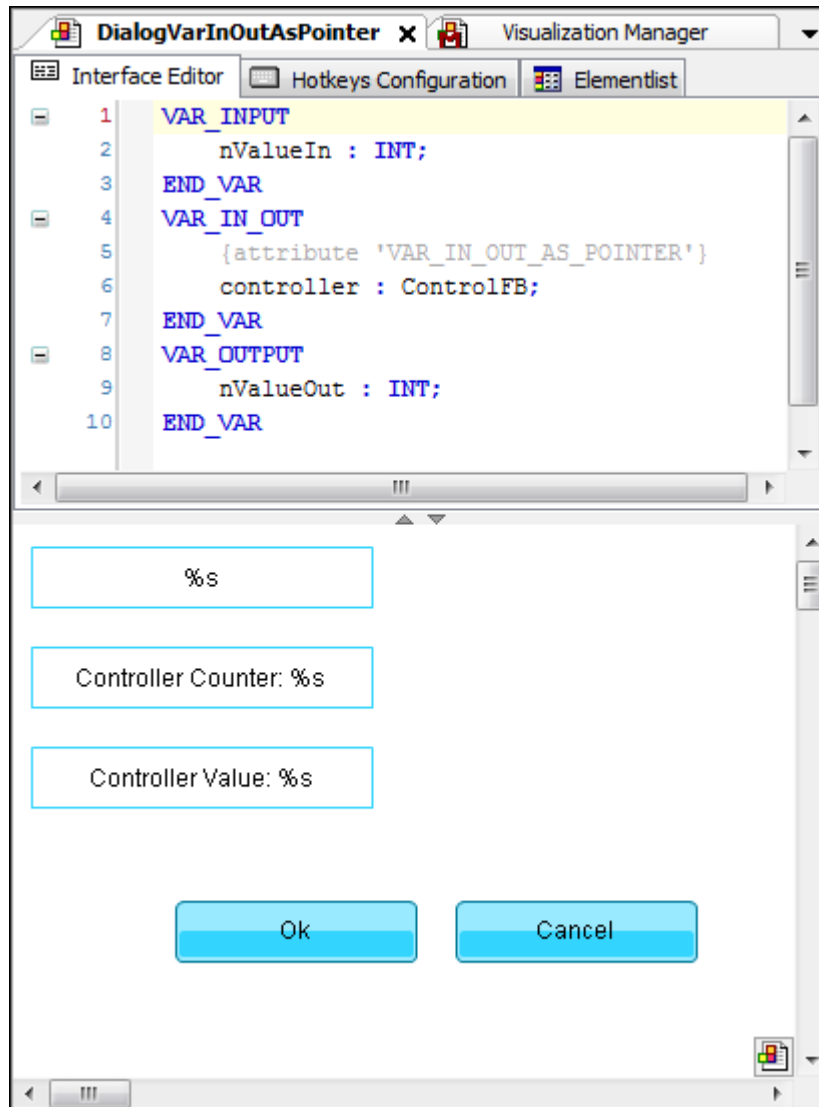


Figure 8-118. Frame Parameter Declaration with VAR_IN_OUT_AS_POINTER

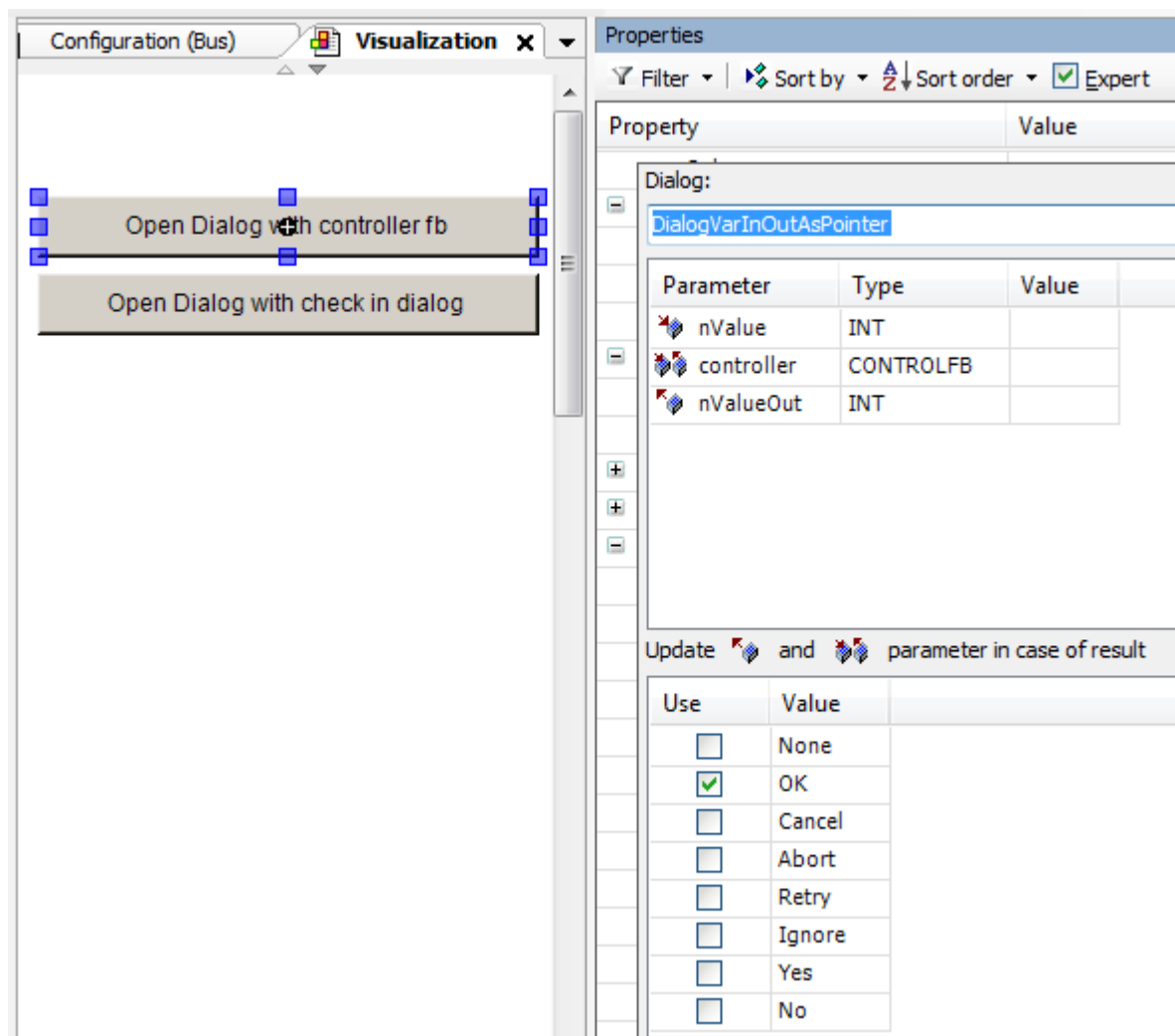


Figure 8-119. User Interface: Dialog Opens

Attribute Parameterstringof

With the help of the pragma {attribute 'parameterstringof'} the instance name of a variable is made accessible by the visualization. Syntax:

```
{attribute 'parameterstringof' := '<variável>'}
```

Example:

In the main program the instance myDUT of the user defined data structure DUT is defined:

```
PROGRAM MainPrg
VAR
    myDUT: DUT;
END_VAR
```

This instance shall be passed to visualization function block Vis (via parameter instance) that is referenced by a frame of another visualization MainVisu:

Elementname	GenElemInst_2
Clipping	<input type="checkbox"/>
Show frame	<input type="checkbox"/>
Scaletype	ANISOTROPIC
References	VisualElem.StructuredTypeNode
References	
PLCWinMT.Application.Vis	
instance	MainPRG.myDut
Position	
v	104

Figure 8-120. Detail of Properties of Visualization Element Contained in MainVisu

In the associated interface editor of Vis we'll find, of course, the input/output variable instance, but also another input variable named instanceStr:

```

1  VAR_INPUT
2  {attribute 'parameterstringof' := 'instance'}
3  instanceStr : STRING;
4  END_VAR
5  VAR_IN_OUT
6  instance : DUT;
7  END_VAR

```

Figure 8-121. Declaration of Parameter Frame with parameterstringof

Although being an input variable instanceStr is not listed within the inputs of the reference (see above picture). This is, because the variable instanceStr carries the attribute 'parameterstringof' and will so be assigned automatically to the name of the variable associated to the attribute. In our example, the associated variable is instance; therefore the string variable instanceStr will get assigned to MainPrg.myDUT and can now be used by the visualization Vis, for example as text variable defining a placeholder '% s'.

Hotkeys Configuration

The Hotkey Configuration editor - to be opened via the **Hotkeys Configuration** command - is available as a separate tab besides the **Basics of Interface Editor** and other editors in the upper part of the visualization editor.

Basics

In addition to the **Keyboard Usage in Online Mode**, this part of the Visualization Editor allows to configure special hotkeys for operating a visualization by keyboard in online mode. That is, an action can be assigned to a certain key or key combination. This configuration is valid exactly for the current visualization. Key configurations, which should be valid within all visualizations of an application, should be done in the **Default Hotkeys**.

It is defined by the device description which keys are supported in a visualization running on that device.

Regard the following sequence of calls when processing key actions:

1. Event handler of the application if activated (optional); e.g. g_VisuEventManager. See **Noticing Events and Input Actions**.
2. Key configuration valid for all visualizations of the application in the **Default Hotkeys**.

3. Definition in **Keyboard Usage in Online Mode**.
4. Special key configurations of the particular visualizations (as handled by the Hotkeys Configuration editor), whereby main visualizations will be regarded before those which are contained in frames.

Regard in this context also the *Hotkey* property of a particular visualization element, which might be configured additionally in the Element Properties editor section **Input Configuration**. Such an element-specific key definition will also be represented in the Hotkeys Configuration editor. It can be edited in both places whereby any modification always will get updated also in the other editor..

Editor

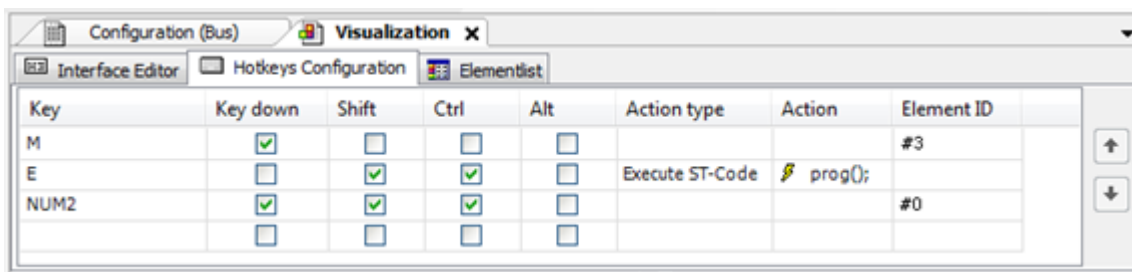


Figure 8-122. Hotkeys Configuration Editor

In each row of the table editor a key or a combination of the key and one/several modifier keys can get assigned to an action. See the following columns:

Key: name of the key. The key name can be entered manually or chosen from the selection list available on a double-click on the field. The list provides all keys defined by the device description, resp. in case of a visualization not directly assigned to a device all keys which are supported by each device.

Key down: if this option is activated, the Action will be executed, when the Key gets pressed. Otherwise the Action will be executed when the Key gets released. If the Action should be executed on Key down and Key up inputs, two appropriate definitions for the Key must be available in the table.

Shift, Ctrl, Alt: if this option is activated, the [Shift] resp. [Ctrl] resp. [Alt] key must be pressed together with the key in order to get the action executed.

Action type: the action type can be chosen from a selection list which is available on a double-click on the field. The action types correspond to those used in the **Element Properties** editor section **Input Configuration** of a visualization element.

Action: exact configuration of the action to be executed; depends on the action type and corresponds to the mouse action like it is used in **Input Configuration** of an element.

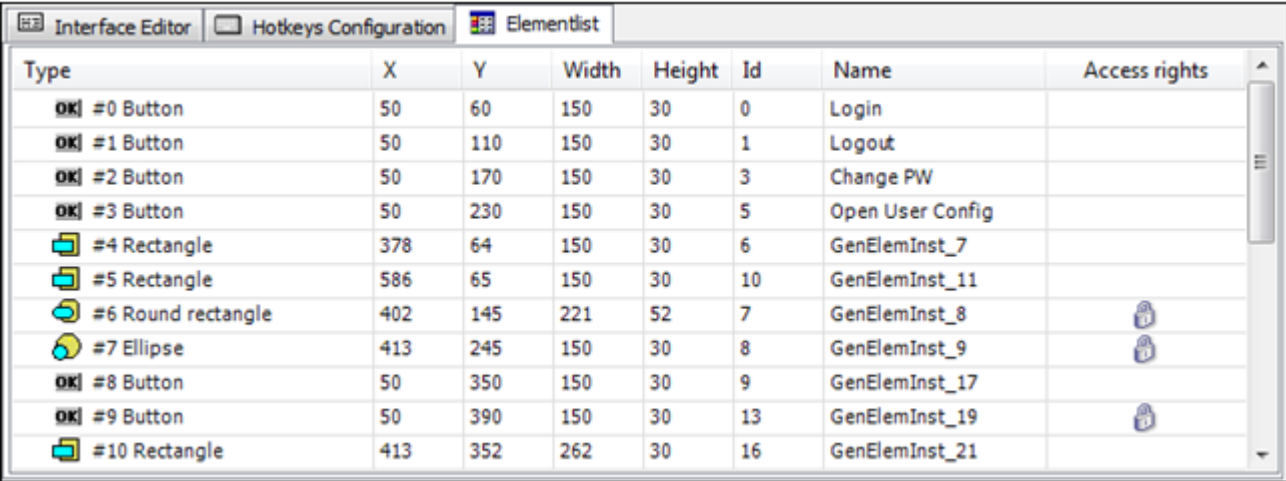
Element ID: ID of the visualization element the Key is assigned to (via the Element Properties editor section Input configuration, Hotkey). Unique identifier within the current visualization.

If a key is associated with several actions, these will be executed in the same order as they are listed top down here in the configuration table. You might change this order by selecting a key definition (click on the table row) and using the arrow buttons right to the table to move the entry up or down.

Element List

This part of the Visualization Editor provides a list of the visualization elements of the currently opened visualization. The Element List can be opened via the **Element List** command (per default on

Visualization menu) and then will be available as a separate tab besides the Interface Editor and other editors in the upper part of the visualization.



Type	X	Y	Width	Height	Id	Name	Access rights
#0 Button	50	60	150	30	0	Login	
#1 Button	50	110	150	30	1	Logout	
#2 Button	50	170	150	30	3	Change PW	
#3 Button	50	230	150	30	5	Open User Config	
#4 Rectangle	378	64	150	30	6	GenElemInst_7	
#5 Rectangle	586	65	150	30	10	GenElemInst_11	
#6 Round rectangle	402	145	221	52	7	GenElemInst_8	
#7 Ellipse	413	245	150	30	8	GenElemInst_9	
#8 Button	50	350	150	30	9	GenElemInst_17	
#9 Button	50	390	150	30	13	GenElemInst_19	
#10 Rectangle	413	352	262	30	16	GenElemInst_21	

Figure 8-123. Lista de Elemento

The elements are listed top down according to their position on the Z-axis of the visualization area, the back-most in the first line. The following values are displayed, but cannot be edited here:

Type: element type and icon as used in the **Toolbox** and element number initially resulting from the sequence of inserting.

X, Y: position of the upper left corner of the element (0,0 = upper left corner of the visualization area).

ID: internally assigned element identifier.

Name: element name as defined in the **Element Properties** editor.

Access rights: if the behavior of an element is limited for some usergroups, this is indicated with the padlock-symbol .

Element(s) can be selected by selecting the respective row(s), whereby the selection always will be synchronized with that in the main visualization editor. Regard however that subelements of a group can only be selected here in the element list.

The position of a selected element on the Z-axis, that is on the background-foreground axis, can be modified by using the following commands:

- **Bring One to Front**
- **Bring to Front**
- **Send One to Back**
- **Send to Back**

The following commands can also be used:

- **Undo**
- **Redo**
- **Delete**

Visualization Object

Properties of a Visualization Object

The dialog *Properties* opens with a click on the command **Element Properties** to be found in the **View Menu** when a visualization object is selected, or in the context menu of a selected visualization object.

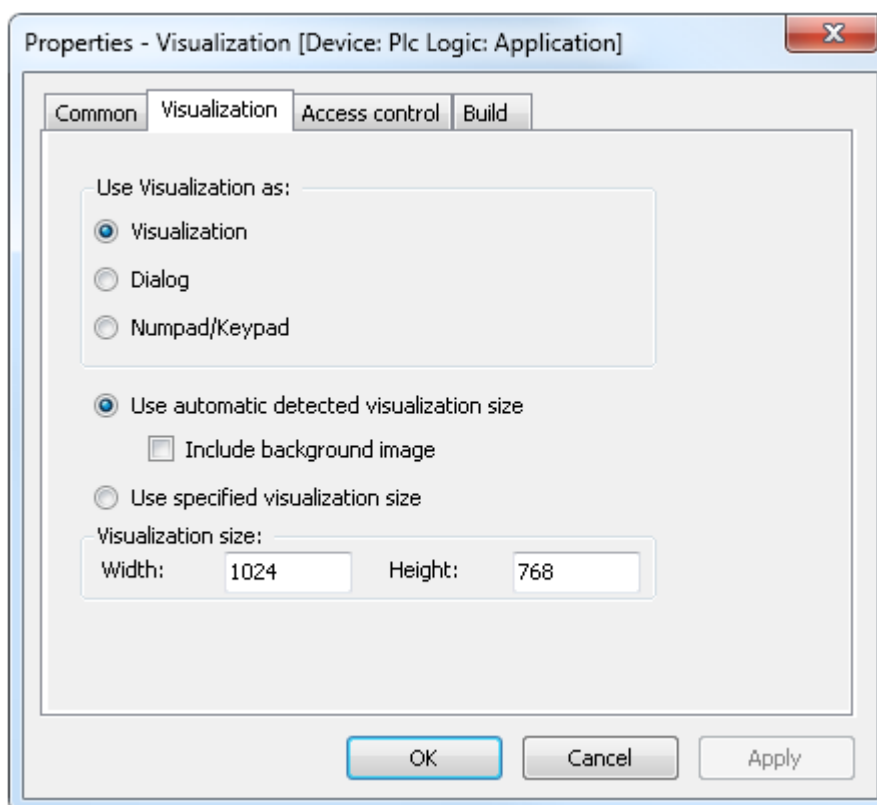


Figure 8-124. 'Properties' Dialog, Category 'Visualization'

For the currently selected visualization object the following properties are displayed and can be modified:

- Use Visualization as:
 - *Visualization*: Visualization is used as a standard visualization object.
 - *Dialog*: the visualization represents a dialog and will be automatically available in other visualizations insofar as it can be specified there in property **Input Configuration > Open Dialog** of a visualization element. See **Input Dialogs** and **Programatic Access on Input Dialog** for further information.
 - *Numpad/Keypad*: the visualization represents a numpad or keypad and will be automatically available in other visualizations insofar as it can be specified there in the **Input Configuration** in category *Write a Variable* (Edit Type) of a visualization element. Regard that the interface of such a numpad/keypad visualization must be identical to that of the standard numpad/keypad provided by the VisuDialogs.library.
- *Use automatic detected visualization size*: The size of the visualization is calculated which exactly includes all currently defined elements and the background image. If the option Include background image however is deactivated, the background image - if exceeding the visualization area defined by the elements - will be cut.

- *Use specified visualization size:* The visualization size is defined by the specified values (Width, Height), regardless whether all visualization elements and the background image will fit into the defined area.
- *Visualization size:* Current *Width* and *Height* of the visualization object (number of pixels)..

Use Cases

Input Dialogs

A visualization can be designed as and declared in its **Properties of a Visualization Object** as a *Dialog*. Dialog visualizations then can be used in other visualizations in order to provide an user input mask. For example they will be offered when you are configuring an *OpenDialog*, *CloseDialog* or *OnDialogClosed* property in category *Input configuration* of a visualization element.

Input and InOut parameters of the dialog visualization have to be defined in the **Interface Editor** and have to be served appropriately by another visualization element which is configured for opening the dialog (**Open Dialog** Configuration). This means that the parameters can be used in the configuration editor for another element.

ATTENTION:

Note that for **Diagnosis Visualization**, means the visualization is running in the programming system, the input parameters must be of scalar type. For usage in Target or Web Visualization also other types (even POU's) might be used.

Besides of possibly self-designed dialogs, standard dialog visualizations are available when library VisuDialogs is included in the project. Currently these are visualizations of a login dialog and of a dialog for opening and saving a file (VisuDialogs.Login, VisuDialogs.FileOpenSave).

Example for Creating and Using an own Dialog Visualization

On a mouse-click on a button a dialog should be opened where credentials settings, in this case user level and password, can be changed. For this purpose we create a visualization named ChangeUserLevel containing this dialog, and another named Visualization containing the calling button.

1. Create the Dialog Visualization ChangeUserLevel

Create a visualization object named ChangeUserLevel and looking like shown in the figure below. In its object properties define that it should be used as a "Dialog". In the current example it is a dialog for entering a user level and a password and will only be closable via the OK button if the given level and password are matching.

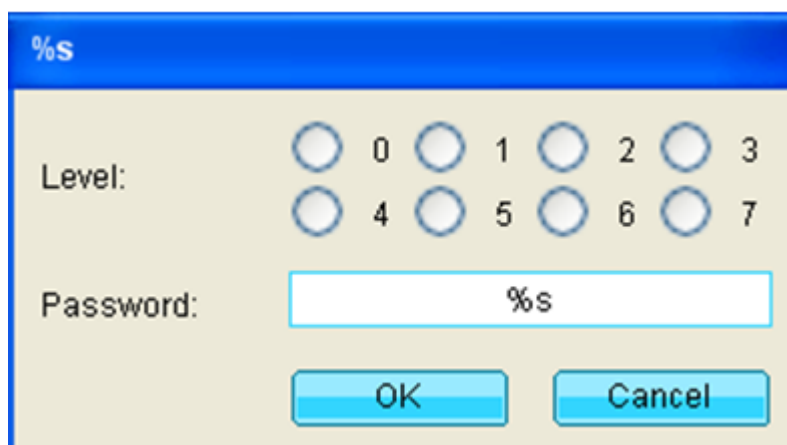


Figure 8-125. ChangeUserLevel Dialog Visualization

The dialog is composed of the following elements:

Element	Description	Element Properties
Image	Background	Static ID: Login; image of an empty dialog with grey background and an empty blue title bar: The image file can be provided by an image pool, where it is stored under the ID Login.
Rectangle	For displaying the dialog name in the title bar	Texts > Text : %s; will be replaced by the value of parameter stTitle, see in the following; Textvariables > Text Variable : stTitle; this variable is defined as input parameter in the interface of the visualization and its value can be defined in the configuration dialog of the visualization element which is used for opening the dialog.
Radiobutton	For entering the user Level	Number of columns: 4 Radiobutton Order: Left to right Radiobutton settings > Radiobutton ; define the 8 areas [0] to [7] (each: Text: <n>, whereby n is 0 to 7) Variable: iLevel; inoutput of the element.
Textfield	For entering the password	Texts > Text : %s Textvariables > Text Variable : stPwd; inoutput of the element. Input configuration: OnMouseDown: Write Variable: Edit type: Edit, Use text output variable
Label	Text Level	Texts > Text : Level:
Label	Text Password	Texts > Text : Password:
Button	Button OK; color indicates whether level and password are matching; mouse-click on button will only be possible if this is the case and will close the dialog	Texts > Text : OK Colors: Color and Alarm color must be different! Color variables > Toggle color : stPwd <> MUX(iLevel, stLevel0, stLevel1, stLevel2, stLevel3, stLevel4, stLevel5, stLevel6, stLevel7); in case password and level are not matching, the color changes to grey. State variables: Deactivate inputs: stPwd <> MUX(iLevel, stLevel0, stLevel1, stLevel2, stLevel3, stLevel4, stLevel5, stLevel6, stLevel7); in case password and level are not matching, any input on the button will not be regarded. Input configuration > OnMouseDown > Close Dialog : ChangeUserLevel, Result: OK; regard that the inputs on the dialog will not be be written to the respective variables when the OK button is used.
Button	Botão Cancel	Colors > Color : = Color of OK-Button Texts > Text : Cancel Input configuration > OnMouseDown > Close Dialog : ChangeUserLevel, Result: Cancel

Table 8-39. Elements of ChangeUserLevel Dialog

Open the **Basics of Interface Editor** of the visualization and declare the following variables the values of the input variables will later be defined in the **Input Configuration, Open Dialog**.

```
VAR_INPUT
    stTitle: STRING;
    stLevel0: STRING;
    stLevel1: STRING;
    stLevel2: STRING;
    stLevel3: STRING;
    stLevel4: STRING;
    stLevel5: STRING;
    stLevel6: STRING;
    stLevel7: STRING;
END_VAR
VAR_IN_OUT
    iLevel: INT;
    stPwd: STRING;
END_VAR
```

2. Create a Visualization “Visualization”

Now a the visualization Visualization with a button for opening the above defined "ChangeUserLevel" dialog and a display of the currently set level is created.

In the current example the following elements are used:

Element	Description	Element Properties
Button	For Opening ChangeUserLevel Dialog	Texts > Text : Change User Level Input configuration > OnMouseDown : Open Dialog; see figure below.
Text field	Level	Texts > Text : %s; title for the dialog. Textvariables > Text Variable : MainPrg.iLevel; assignment of level input and program variable.

Table 8-40. Elements of Visualization Dialog

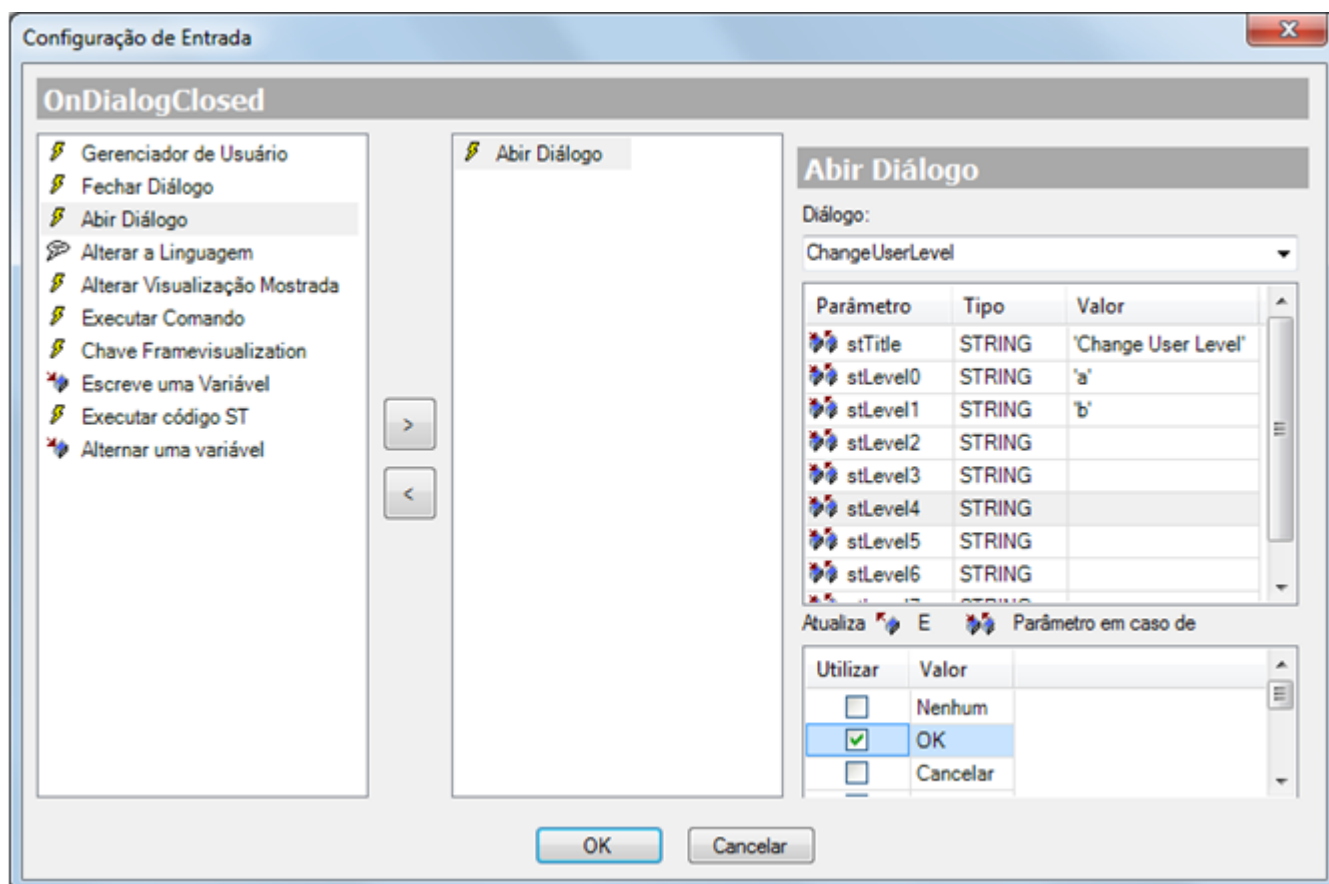


Figure 8-126. Input Configuration of the Button which Should open the Self-Defined Dialog Visualization
ChangeUserLevel

Here the dialog to be opened on a mouse-click on the button must be defined: For this purpose choose *ChangeUserLevel* from the selection list below label *Dialog:*. The input (VAR_INPUT) and inout (VAR_IN_OUT) parameters of the dialog visualization (interface of *ChangeUserLevel*) will be listed and you can define, which values should be read by the *ChangeUserLevel* dialog each time it gets opened. Regard for the current example - as shown in the figure above- only two Level-Password combinations are configured: If the user selects Level 0, he must insert a as password in order to get the OK button enabled. If he selects Level 1, he must insert b as password. Selecting one of the other levels, currently no password is needed.

Further define on which action in the *ChangeUserLevel* dialog this dialog should get closed and its outputs and inouts should get written. In our example set a check at the *Use of OK*. So later on a mouseclick on the OK button the *ChangeUserLevel* dialog will be closed and the variables *iLevel* and *stPwd* (VAR_IN_OUT) will be written with the values resulting from the user input.

See for further information on the *OpenDialog* configuration dialog **Open Dialog**.

3. Running “Visualization” in Online Mode

When you now load and start the visualization *Visualization* in online mode (possible as Target-Visualization, web-visualization or *diagnosis visualization*), you can open the *ChangeUserLevel* dialog via a mouse-click on the designed button in *Visualization*:

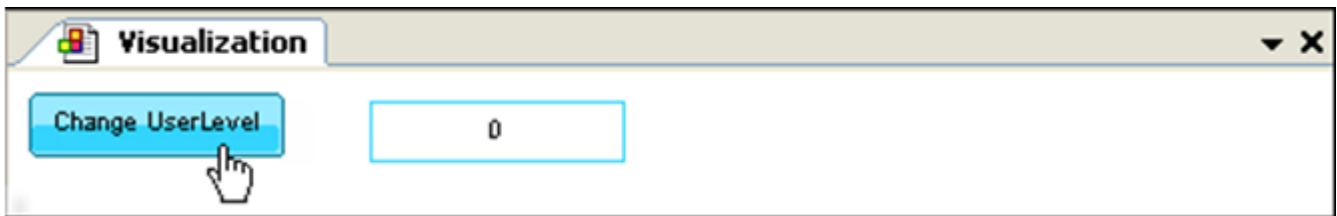


Figure 8-127. Run "Visualization", click on button Change UserLevel

The dialog opens. Select a level, e.g. 1 (will be written to iLevel) and enter the matching password b (will be written to stPwd). These values will be compared by the OK button; as long as the values don't match, the button is grey and not available for input:

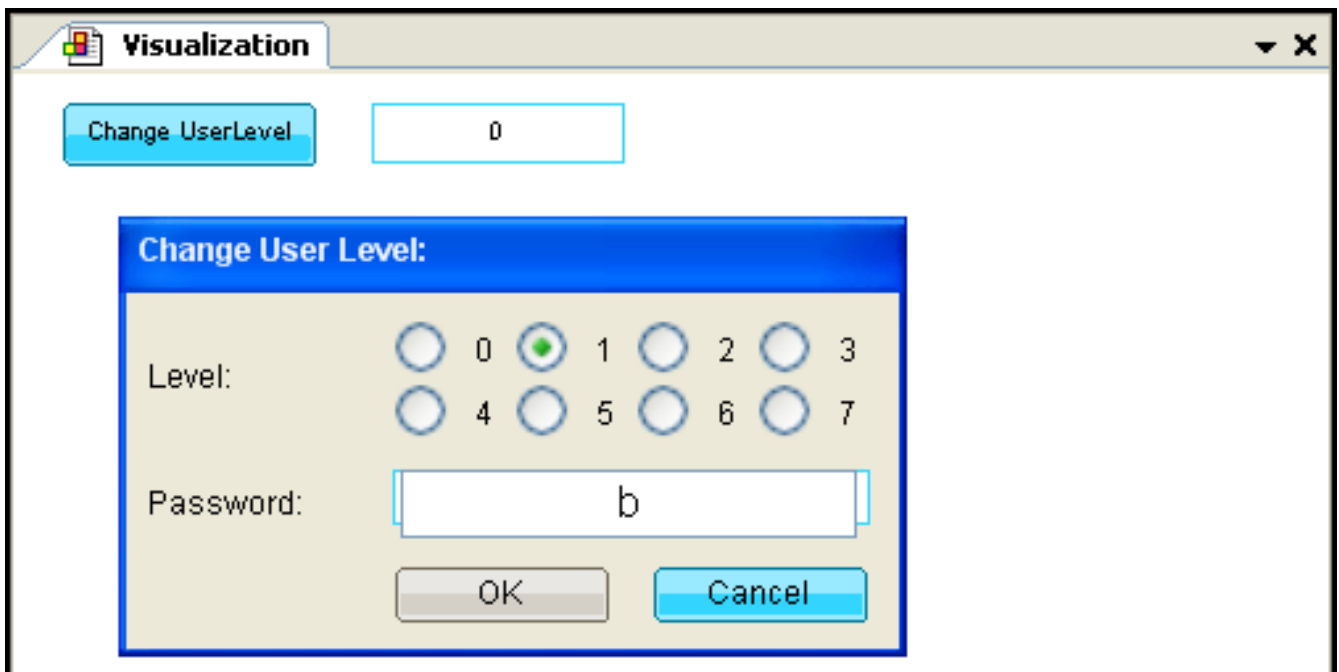


Figure 8-128. Inputs in Change User Level Dialog

You have inserted matching values, so button OK changes color to blue and you can click on it to close the dialog.

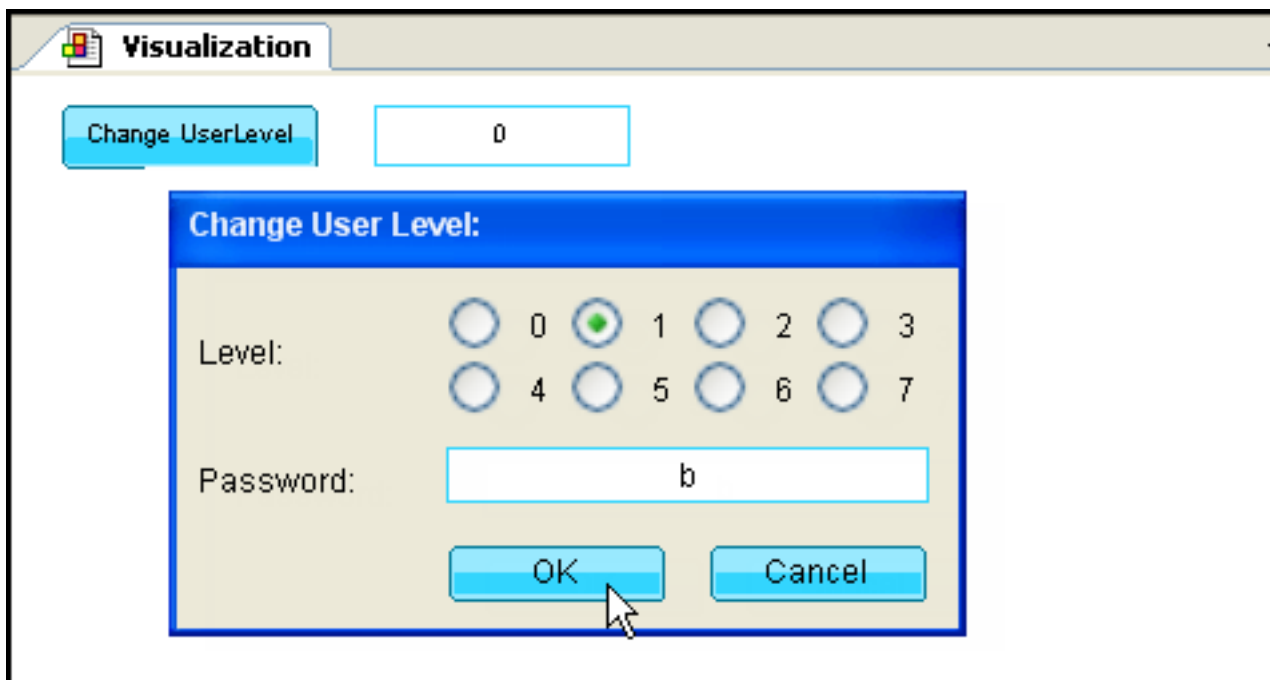


Figure 8-129. OK available for closing the dialog

The values now get really written (Note that up to now they just have been kept on the stack) and the changed level (1) is displayed.

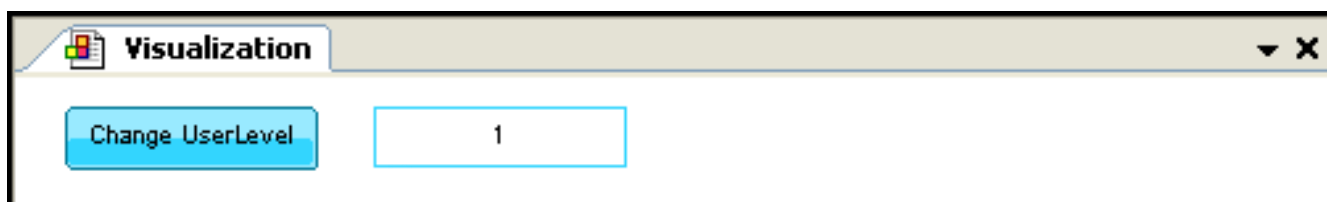


Figure 8-130. New value of user level is displayed

Programatic Access on Input Dialog

Programmatic access on a **Input Dialogs** created in the visualization is possible:

The parameters defined in the (**Basics of Interface Editor**) of a dialog visualization automatically get written to a structure named `<dialogname>_VISU_STRUCT` ("dialogname" = name of the visualization). This structure might be accessed by the application, for example via a function, which is configured to be called in case of a user input on a visualization element. This allows to program the opening of a dialog in a visualization and also the reaction on any user inputs on this dialog.

DialogManager

All visualizations declared as dialog will be instantiated automatically and managed by the internal DialogManager. This manager can be accessed via the also internal VisuManager by calling method `GetDialogManager()`. The DialogManager provides some methods for handling a visualization dialog by the application.

The available methods are described in **Methods for the DialogManager**.

Example of Using the Login-Dialog Provided by VisuDialog.library

In this example a button element in a visualization is configured in a way that on a mouse-click on this button a login dialog gets opened, where a user name and password can be entered. A default login dialog is used for this purpose, provided by the VisuDialog.library. However, self-created dialogs could be handled in the same way:



Figure 8-131. Login Dialog

Configuration of the Login Button

A button element is inserted in a visualization. It gets configured with a text property "Login" and the following two input properties:

1. OnMouseDown / Execute ST-Code: call function `OpenLoginDialog(pClientData)`. Variable `pClientData` must be passed to an appropriate function `OpenLoginDialog` available in the project.

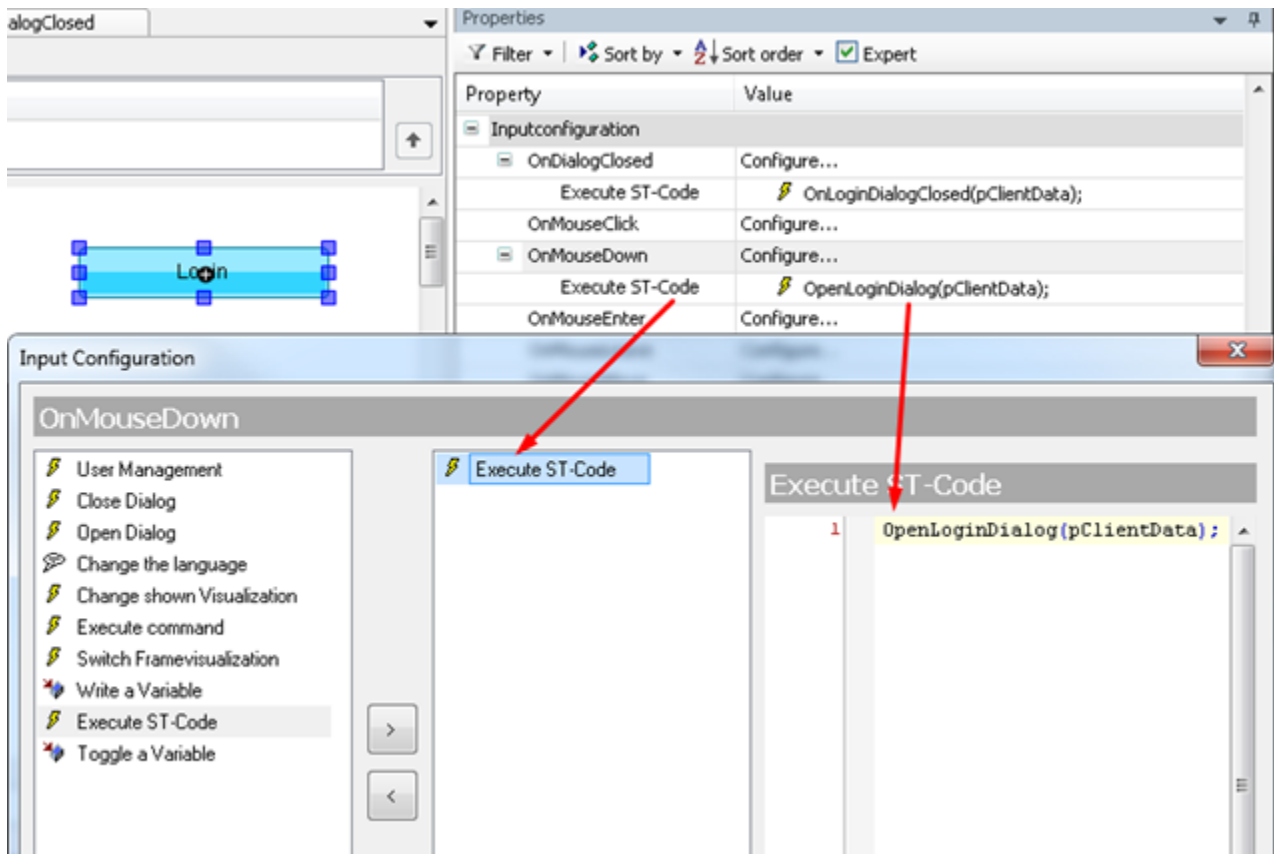


Figure 8-132. Function Call 'OpenLoginDialog()'

2. OnDialogClosed / Executar código ST: call function OnLoginDialogClosed(pClientData)

The respective functions OpenLoginDialog and OnLoginDialogClosed are available as POU's in the project. See in the following a possible implementation.

Implementation of Functions OpenLoginDialog()

Function OpenLoginDialog() getting called when the dialog gets opened., that is on a OnMouseDown action on the button element. It reads the parameters of the opened dialog.

```

FUNCTION OpenLoginDialog : bool
VAR_INPUT
    pClientData : POINTER TO VisuElems.VisuStructClientData;
END_VAR
VAR
    dialogMan : VisuElems.IDialogManager;
    messageDialog : VisuElems.IVisualisationDialog;
    result : VisuElems.Visu_DialogResult;
    stTitle : STRING := 'Message ...';
    stPasswordLabelText: STRING;
    stUserLabelText: STRING;
    stUsername: STRING;
    myRect : VisuElems.VisuStructSimpleRectangle;

    pTempClient : POINTER TO VisuElems.VisuStructClientData;
    iteratorSave : INT;
END_VAR

// Open the dialog on all clients

```



```

dialogMan := VisuElems.g_VisuManager.GetDialogManager(); // the
DialogManager is provided via the implicitly available VisuManager
IF dialogMan < > 0 AND pClientData < > 0 THEN

    messageDialog := dialogMan.GetDialog('GlobalMessageDialog'); // dialog
    to be handled is specified
    IF messageDialog < > 0 THEN
        myRect.ptTopLeft.iX := PLC_PRG.xPos;
        myRect.ptTopLeft.iY := PLC_PRG.yPos;
        myRect.ptBottomRight.iX := 10;
        myRect.ptBottomRight.iY := 10;
        dialogMan.OpenDialog3(messageDialog, pClientData, TRUE, ADR(myRect),
        0, VisuElems.Visu_InputFlags.GlobalOpenCloseDialog);
    END_IF
END_IF

```

Implementation of Functions OnLoginDialogClosed()

OnLoginDialogClosed() defining the reaction on a closing of the dialog.

```

FUNCTION OnLoginDialogClosed : bool
VAR_INPUT
    pClientData : POINTER TO VisuElems.VisuStructClientData;
END_VAR
VAR
    dialogMan : VisuElems.IDialogManager;
    messageDialog : VisuElems.IVisualisationDialog;
    result : VisuElems.Visu_DialogResult;
    itfDialogManager2 : VisuElems.IDialogManager2 := 0;
END_VAR

dialogMan := VisuElems.g_VisuManager.GetDialogManager(); // the
DialogManager is provided via the implicitly available VisuManager
IF dialogMan < > 0 AND pClientData < > 0 THEN

    messageDialog := dialogMan.GetDialog('GlobalMessageDialog'); // dialog
    to be handled is specified
    IF messageDialog < > 0 THEN
        IF (itfDialogManager2 = 0) THEN
            __QUERYINTERFACE(VisuElems.g_VisuManager.GetDialogManager(),
            __itfDialogManager2);
        END_IF
        IF (itfDialogManager2 < > 0) THEN
            __itfDialogManager2.CloseDialog2(messageDialog, pClientData,
            __VisuElems.Visu_InputFlags.GlobalOpenCloseDialog);
        ELSE
            __dialogMan.CloseDialog(messageDialog, pClientData);
        END_IF
    END_IF
END_IF

```

Methods for the DialogManager

GetDialog (STRING stName): Provides the dialog, which is currently concerned by an event, as **IVisualisationDialog**:

MÉTODO **GetDialog**

Parâmetros:

Nome	Tipo	Comentário
GetDialog	VAR_OUTPUT	
stName	VAR_INPUT	

Figure 8-133. Parameters of Method GetDialog

GetClientInterface():Returns a pointer on the dialog structure.

MÉTODO **GetClientInterface**

Parâmetros:

Nome	Tipo	Comentário
GetClientInterface	VAR_OUTPUT	
dialog	VAR_INPUT	
pClient	VAR_INPUT	

Figure 8-134. Parameters of Method GetClientInterface

OpenDialog():Opens the dialog for the given client.

MÉTODO **OpenDialog**

Parâmetros:

Nome	Tipo	Comentário
OpenDialog	VAR_OUTPUT	
dialog	VAR_INPUT	
pClient	VAR_INPUT	
bModal	VAR_INPUT	
pRect	VAR_INPUT	

Figure 8-135. Parameters of Method OpenDialog

CloseDialog(): Closes the dialog for the given client.

MÉTODO CloseDialog

Parâmetros:

Nome	Tipo	Comentário
CloseDialog	VAR_OUTPUT	
dialog	VAR_INPUT	
pClient	VAR_INPUT	

Figure 8-136. Parameters of Method CloseDialog

Noticing Events and Input Actions

Methods provided by **Visualization Libraries** might be used in the application to get notified on certain events caused by an user input.

Notice Closing of Edit Controls (Writing a Variable)

The application can notice when an 'edit control' is closed in an assigned visualization, that means when a variable gets written by an user input in an edit control element. This might be useful, if an additional action should be executed after an edit control has been closed.

To get notified about such events, a function block instance (implementing the interface `VisuElems.IEditBoxInputHandler`, provided by `VisuElemBase.library`) can be attached to the global instance `VisuElems.Visu_Globals.g_VisuParamEventManager`, using method `SetEditBoxEventHandler` of this instance like shown in the example below.

`VisuElems.IEditBoxInputHandler` requires method `VariableWritten`, which will be called after a variable has been written by an user input in a visualization.

Example

There is a visualization with the following elements:

1. Two rectangles, each with an edit control field (input "Write a variable") where the user can enter a value for variable i or j.

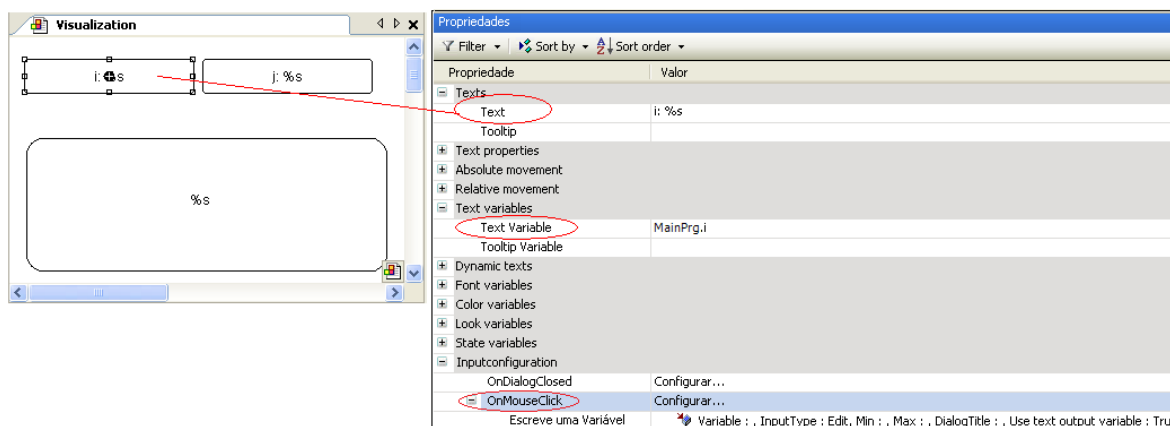


Figure 8-137. Input Elements

2. Another rectangle for displaying a string (Text Variable `stInfo`) with some information when one of the edit fields in the upper two rectangles gets closed, that means when a value gets written.

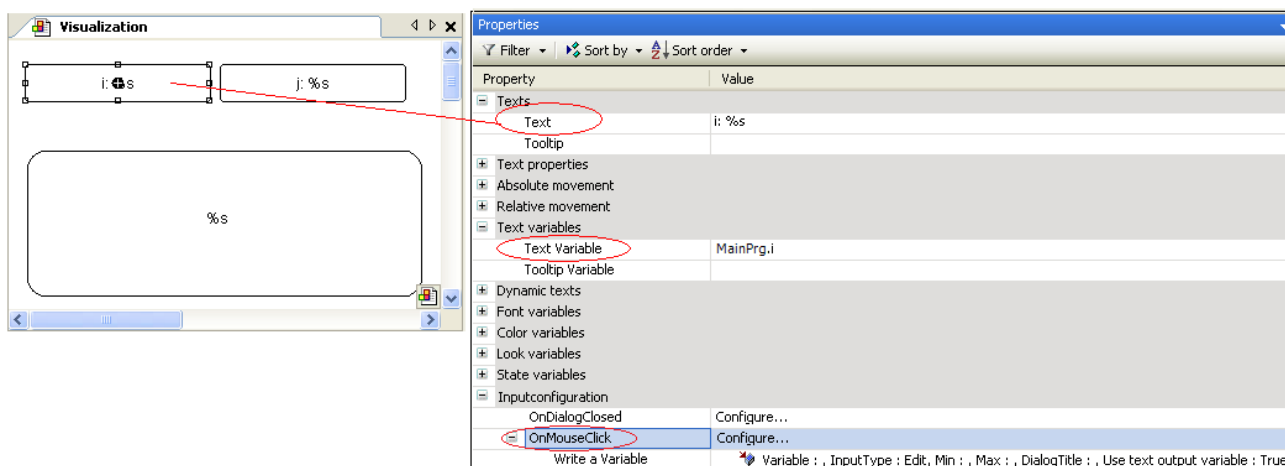


Figure 8-138. String Display Element

The application contains the following POU:

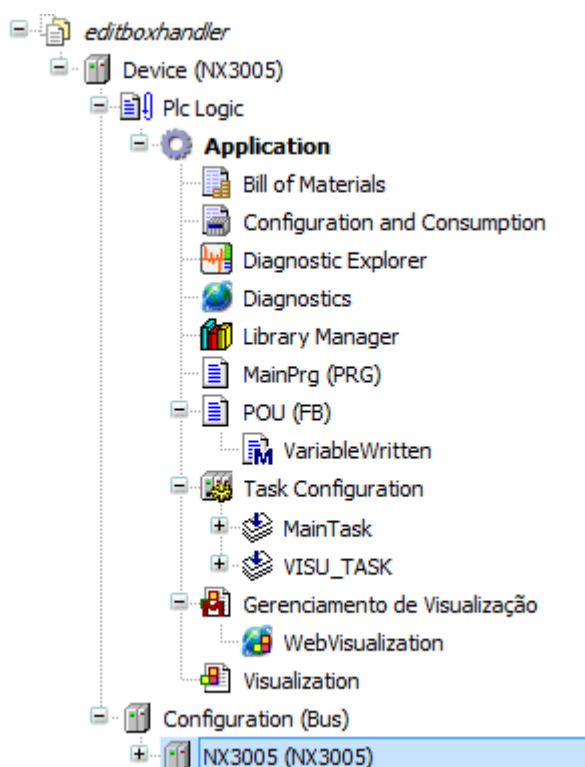


Figure 8-139. Aplicação com POU

Programm MainPrg

```

PROGRAM MainPrg
VAR_INPUT
    i:INT; (* variable to be written by user input in visualization *)
    j:REAL; (* variable to be written by user input in visualization *)
    stInfo : STRING; (* information on the user input via the edit control
    field;
    string gets composed by method VariableWritten; result is displayed in
    the lower rectangle of the visualization *)

```

```

END_VAR
VAR
    inst : POU;
    bFirst : BOOL := TRUE;
END_VAR

IF bFirst THEN
    bFirst := FALSE;
    VisuElems.Visu_Globals.g_VisuEventManager.SetEditBoxEventHandler(inst);
    (* Call of method VariableWritten *)
END_IF

```

FUNCTION_BLOCK POU

FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IEditBoxInputHandler

(no further declarations, no implementation code)

Method 'VariableWritten', Assigned to POU

```

METHOD VariableWritten : BOOL
(* provides some information always when an edit control field is closed
in the visualization, that is a variable gets written by user input in one
of the upper rectangles *)
VAR_INPUT
    pVar : POINTER TO BYTE;
    varType : VisuElems.Visu_Types;
    iMaxSize : INT;
    pClient : POINTER TO VisuElems.VisuStructClientData;
END_VAR

// String stinfo, which will be displayed in the lower rectangle, is
composed here
MainPrg.stInfo := 'Variable written; type: ';
MainPrg.stInfo := CONCAT(MainPrg.stInfo, INT_TO_STRING(varType));
MainPrg.stInfo := CONCAT(MainPrg.stInfo, ', adr: ');
MainPrg.stInfo := CONCAT(MainPrg.stInfo, DWORD_TO_STRING(pVar));
MainPrg.stInfo := CONCAT(MainPrg.stInfo, ', by: ');
MainPrg.stInfo := CONCAT(MainPrg.stInfo,
SEL(pClient^.globaldata.clienttype =
VisuElems.Visu_ClientType.Webvisualization, 'other visu', 'webvisu'));

```

Notice Key Events

The application can notice when a key event is released in an assigned visualization, that means when the user operates and releases a key when the visualization is active. Regard the options of configuring **Keyboard Usage in Online Mode** of visualizations.

To get notified about such events, a function block (implementing the interface VisuElems.IVisuUserEventManager, provided by VisuElemBase.library) can be attached to the global instance VisuElems.Visu_Globals.g_VisuEventManager, using the method SetKeyEventHandler of this instance like in the following example.

Example

There is a visualization with a visualization element that displays the value of MainPrg.stInfo as a text string.

That means that in online mode in this element the following information will be shown on the currently released key action: "KeyEvent up: <dwKey>, key: <key id>, modifier: <dwModifiers>, by: <pClient^.globaldata.clienttype.....>".

Example: "KeyEvent up: TRUE, key: 75, modifier: 0, by: webvisu".

Property	Value
ElementName	GenElemInst_2
Type of element	VISU_ST_ROUNDRE
+ Position	
+ Center	
+ Colors	
+ Elementlook	
- Texts	
Text	%s
Tooltip	
+ Text properties	
+ Absolute movement	
+ Relative movement	
- Text variables	
Text Variable	MainPrg.stInfo
Tooltip Variable	

Figure 8-140. Text configuration of the visualization element

The application contains the following POUs:

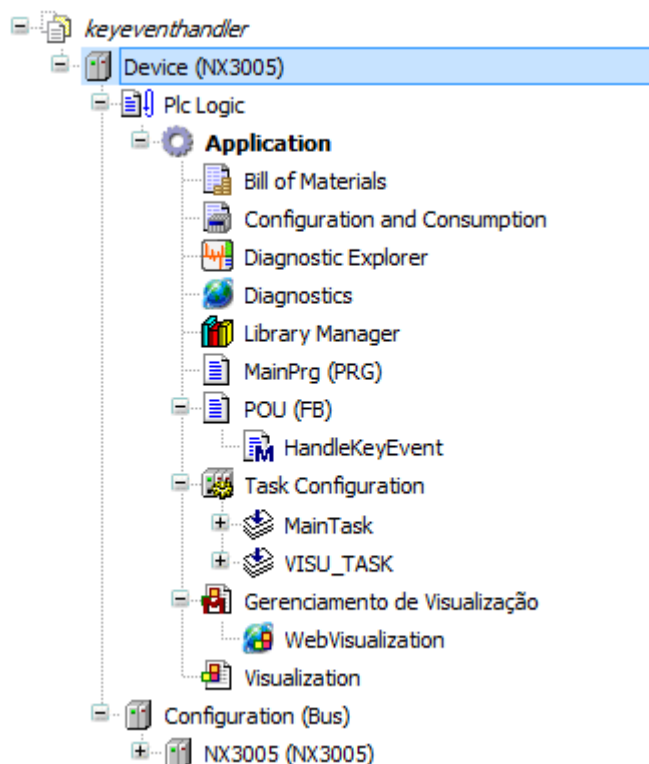


Figure 8-141. Application with POUs

Programm MainPrg

```

PROGRAM MainPrg
VAR_INPUT
    stInfo : STRING;
END_VAR
VAR
    inst : POU;
    bFirst : BOOL := TRUE;
END_VAR

IF bFirst THEN
    bFirst := FALSE;
    VisuElems.Visu_Globals.g_VisuEventManager.SetKeyEventHandler(inst);
END_IF

```

This allows to get notification when a key event is released.

FUNCTION_BLOCK POU

FUNCTION_BLOCK POU IMPLEMENTS VisuElems.IKeyEventHandler

No further declarations, no implementation code!

Method HandleKeyEvent, Atribuído a POU

The interface VisuElems.IVisuUserEventManager requires a method with the following signature that will be called with all available information.

```

/// This method will be called after a key event is released.
/// RETURN:
/// TRUE - When the handler has handled this event and it should not be
handled by someone else
/// FALSE - When the event is not handled by this handler
METHOD HandleKeyEvent : BOOL
VAR_INPUT
    /// Event type. The value is true if a key-up event was released.
    bKeyUpEvent : BOOL;
    /// Key code
    dwKey : DWORD;
    /// Modifier. Possible values:
    /// VISU_KEYMOD_SHIFT : DWORD := 1;
    /// VISU_KEYMOD_ALT : DWORD := 2;
    /// VISU_KEYMOD_CTRL : DWORD := 4;
    dwModifiers : DWORD;
    /// Pointer to the client structure where the event was released
    pClient : POINTER TO VisuElems.VisuStructClientData;
END_VAR
VAR
END_VAR

MainPrg.stInfo := 'KeyEvent up: ';
MainPrg.stInfo := CONCAT(MainPrg.stInfo, BOOL_TO_STRING(bKeyUpEvent));
MainPrg.stInfo := CONCAT(MainPrg.stInfo, ', key: ');
MainPrg.stInfo := CONCAT(MainPrg.stInfo, DWORD_TO_STRING(dwKey));
MainPrg.stInfo := CONCAT(MainPrg.stInfo, ', modifier: ');
MainPrg.stInfo := CONCAT(MainPrg.stInfo, DWORD_TO_STRING(dwModifiers));
MainPrg.stInfo := CONCAT(MainPrg.stInfo, ', by: ');
MainPrg.stInfo := CONCAT(MainPrg.stInfo,
SEL(pClient^.globaldata.clienttype =
VisuElems.Visu_ClientType.Webvisualization, 'other visu', 'webvisu'));

```

Visualization Elements

What can be Done with a Visualization Element?

User Inputs When Editing an Element

In the Visualization editor, different user inputs (mouse commands or menu commands) can be done due to the position of the cursor in the editor and the state of the element. If the cursor position allows user inputs, the cursor changes its shape in order to signal this to the user.

An element can be in the state not selected or selected. A selected element is visualized with additional little black squares, that represents the position and size of it and used commands are affecting on this element.

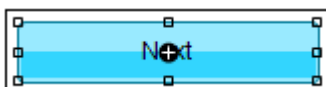


Figure 8-142. Selected Button

Commands

Due to the element and the selection in the editor, different commands are available and not grayed. General information about commands can be found under **Visualization Commands**.

Edit a Visualization Element




Editor Action	Reaction of the Element Properties
With focus on an element and a hand-symbol cursor, a click on the element will select it and therefore the position and size squares of the element are visible on the outline.	The properties of the currently selected elements will be immediately displayed in the Properties view and can be edited there.
With focus on an element and an input cursor  , a click on the element will open an inline editor. There a name/text can be entered. Close the inline editor by [Enter] .	Text under category Texts contains the entered text
With focus on the center of an selected element, the cursor shape changed to  and the center can be dragged and dropped to the desired X-/Y-position.	X e Y under category Center
With focus on a selected element, the cursor shape changed to  and the element can be dragged and dropped to the desired X-/Y-position.	X e Y under category Position
<p>The element is selected. With focus on one of the little squares on the element outline, position and size can be dragged and dropped into the desired size. Resizing an element in this way will reset the center.</p> <p>By pressing simultaneously the [Shift] key while moving the mouse the width and height of the element are scaled proportionately</p> <p>By pressing simultaneously the [Ctrl] key while moving the mouse the width and the height of the element is changed symmetrically to the center.</p> <p>By pressing simultaneously the [Shift] key and the [Ctrl] key while moving the mouse the width and the height of the element is changed symmetrical to the center and the width and the height are scaled proportionately.</p>	<p>Due to the element type, position and size is stored in different ways under category Position:</p> <p>X, Y</p> <p>X, Y, Width, Height.</p> <p>category Points [0] .. [<n>]: X, Y</p>

Table 8-41. Edit a Visualization Element

Multiple Selection

Multiple selection is possible by:

- keeping [Shift] pressed when selecting/deselecting the particular elements.
- drawing a rectangle around multiple elements with the left mouse-button pressed.
- using **Select All**.

Deselection can be done by **Select None**. Alternatively elements can be selected in the **Element List** with help of [Shift] and [Ctrl] . If multiple elements are selected, property modifications in the *Properties* window will be applied to all.

Shift Selection by Keyboard Usage

If an element is selected, the selection can be shifted to the next element according to the insertion order (z-order) by using the [Tabulator] key. Using [Shift] + [Tabulator] the selection is shifted to the previous one..

Group Elements

Multiple selected elements can be grouped with help of command **Group**. Then the selected elements are fixed in a group. Its subelements can only be selected via **Element List**. With **Ungroup** the group is broken down into its elements.

Change the Order and Align of Elements

To define where an element should be positioned on the Z-axis of the visualization area, that is in back- or foreground or somewhere in between, use the commands by default available in *Visualization*, **Order**.

To align two or several elements use the commands which are by default in the *Visualization* submenu, **Alignment**.

Text and Language in Visualization

Text can be assigned to a visualization element by an appropriate definition in the Element Properties editor or directly ("inline") in the Visualization editor. For the latter, a text input field can be opened by pressing [Space] . See also **User Inputs When Editing an Element**.

Text lists are used to manage the texts. A text list is a POU containing text strings which are uniquely (within the project) referenced by the combination of the textlist name and the list-internal text string ID. Different language versions of a text can be defined there and so each text additionally is marked by a language identifier. At least a default language is defined for each text.

See under chapter **Menu Text List** for detailed information on handling text lists.

Directories providing textlist files for the usage in a visualization are specified in **Project Settings**, Category **Visualization**.

Text Types

In the properties of a visualization element you specify a textlist name and an ID (string) and thus determine which text should be displayed in the element in online mode. You can additionally influence the text display by using **Formatting Text**. Due to the fact that the ID can be specified via a variable, a dynamic switching of texts is possible.

If the currently used textlist(s) provide several language versions of a text, the language to be used in the visualization can be switched (language switching) by the help of visualization elements which are configured for that purpose. That means they must have an input property defining a **Change the Language** mouse-action (**Input Configuration**).

Thus dynamic use and multi-language support is possible for visualization texts. Basically there are two types of text in a visualization:

- **Static texts:** Static text assigned to a visualization element cannot be changed in online mode (just the language, if multiple language versions are available in the textlist, always will be adapted to the language currently used in the visualization). Static text is defined in category *Texts* in the properties dialog of the visualization element and is managed in an automatically created textlist named *GlobalTextList*.
- **Dynamic texts:** Dynamic text can be changed in online mode by the input of a variable or a user input. It is managed in Textlists which must be created by the user. Dynamic text is assigned to a visualization element in category *Dynamic texts* in the properties dialog of the element. There one of the available dynamic textlists as well as the ID of the desired text are to be specified..

Formatting Text

Besides the input of a pure text string you also can use formatting sequences to determine the text display in online mode. A formatting sequence always consists of a "%" followed by a character. It can be used singly or in combination with a pure text string.

If you include "%s" in a text string, then this location in online mode will be replaced by the value of the variable which is specified in the 'Text variables' property 'Text variable' (text output variable). If you want to display the instance name of a variable being passed to a visualization function block, you must use the pragma **Attribute Parameterstringof**

Notice that you can use any formatting string conforming with the standard C-library function 'sprintf', you just have to make sure that it fits to the type of the used variable.

Sample List of Strings and Arguments

Character after %	Argument/Output as
d, i	Decimal number
b	Binary number
o	Unsigned octal number (without leading zero)
x	Unsigned hexadecimal number (without leading 0x)
u	Unsigned decimal number
c	Single character
s	String: this location in online mode will be replaced by the value of the variable which is specified in the 'Text variables' property 'Text variable'.
f	REAL values Syntax: % <alignment><minimal width><accuracy>] The alignment is defined by a minus-sign (left aligned) or a plus-sign (right aligned, default); accuracy defines the number of places behind the comma (default: 6); see example below.

Table 8-42. Formatting Arguments

Example

Entry in property field *Text*:

```
Fill level %2.5f mm
```

Entry in field *Text variable* (variable of type REAL, providing the fill level value):

```
MainPrg.fvar1
```

Output in online mode:

```
Fill level 32.89999 mm
```

If you want to get displayed a percent sign % combined with one of the formatting strings mentioned above, you must enter "% %". For example: Enter "Rate in % %: %s" to get displayed in online mode "Rate in %: 12" (if the text display variable currently is "12").

The font to be used for the online display of the text also can be defined in the Element Properties editor of the visualization element. See in this concern the category **Texts**.

The horizontal and vertical alignment of a text is defined in the Element Properties editor of the visualization element.

System Time Output

If you enter "%t", followed by a sequence of special placeholders in squared brackets, then in online mode this location will be replaced by the system time. The placeholders define the display format, see the table below.

Valid Placeholders	Description
ddd	Name of the weekday, abbreviated, for example "Wed"
dddd	Name of the weekday, for example "Wednesday"
dddddd	Weekday as decimal number (0 – 6; Sunday is 0)
MMM	Name of the month, abbreviated, for example "Feb"
MMMM	Name of the month, for example "February"
d	Day of month as number (1 – 31), for example "8"
dd	Day of month as number (01 – 31), for example "08"
M	Month as number (1 – 12), for example "4"
MM	Month as number (01 – 12), for example "04"
jjj	Day of the year as number (001-366), for example "067"
y	Year without specifying the century (0-99), for example "9"
yy	Year without specifying the century (00-99), for example "09"
yyy	Year with specifying the century, for example "2009"
HH	Hour, 24-hours format (01-24), for example "16"
hh	Hour, 12-hours format (01-12), for example "8" for 16 o'clock
m	Minutes (0-59), without preceded null, for example "6"
mm	Minutes (00-59), with preceded null, for example "06"
s	Seconds (0-59), without preceded null, for example "6"
ss	Seconds (00-59), with preceded null, for example "06"
ms	Milliseconds (0-999), without preceded null, for example "322"
t	Identifier for the display in 12-hours format: A (hours <12) resp. P (hours >12), for example "A" in case of 9 o'clock in the morning
tt	Identifier for the display in 12-hours format: AM (hours <12) resp. PM (hours >12), for example "AM" in case of 9 o'clock in the morning
''	Text strings containing one of the above listed placeholders must be included in single quotation marks; all other texts within the format string can be used without quotation marks; for example 'update', because it contains "d" and "t"

Table 8-43. Valid Placeholders

Example

```
%t['Last update:' ddd MMM dd.MM.yy 'at' HH:mm:ss]
```

Display in online mode:

```
Last update: Wed Aug 28.08.02 at 16:32:45
```

Input Configuration

One or several of the after-actions described in the following can be defined in dialog *Input Configuration* for a certain event (OnMouse..., tap, toggle) on a element. The dialog opens on a mouse-click on the value field in the *Properties* window of that event. The name of the event the after-actions will be selected for are visible as title in the grey line.

User Management

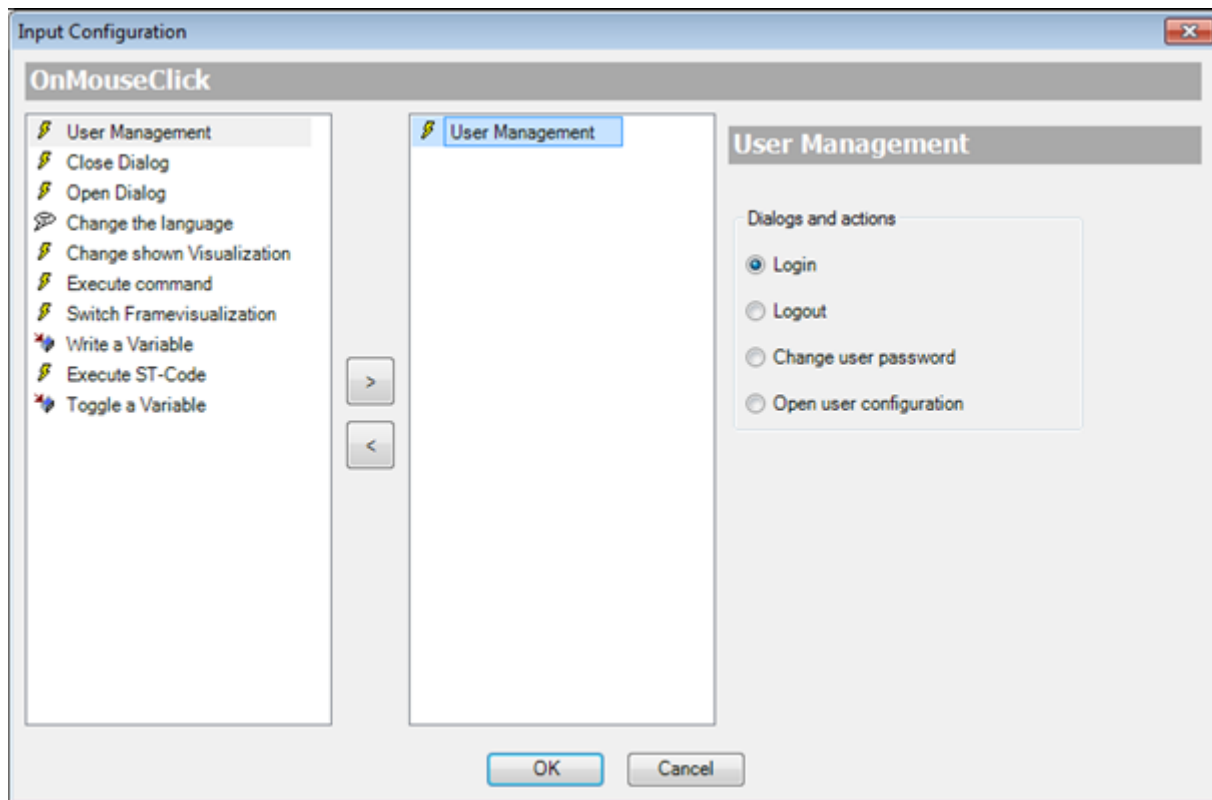
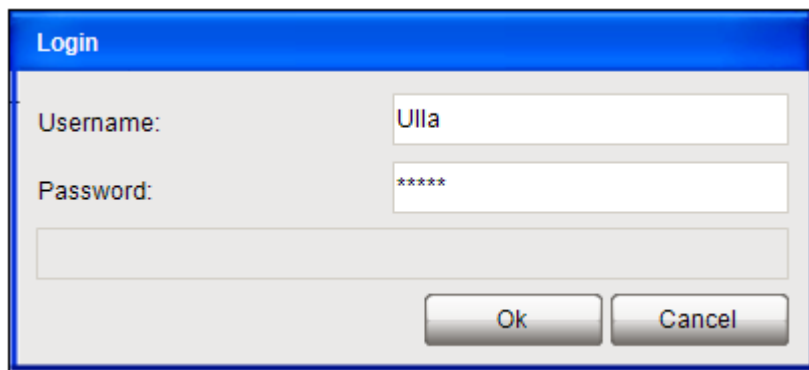


Figure 8-143. User Management

Here you can define that dialogs or actions run on the selected event. It allows the visualization user in online mode to activate one of the listed user interfaces/actions like it is provided in VisuUserManagement. The dialogs are listed in tab *Visualizations* when they are available.

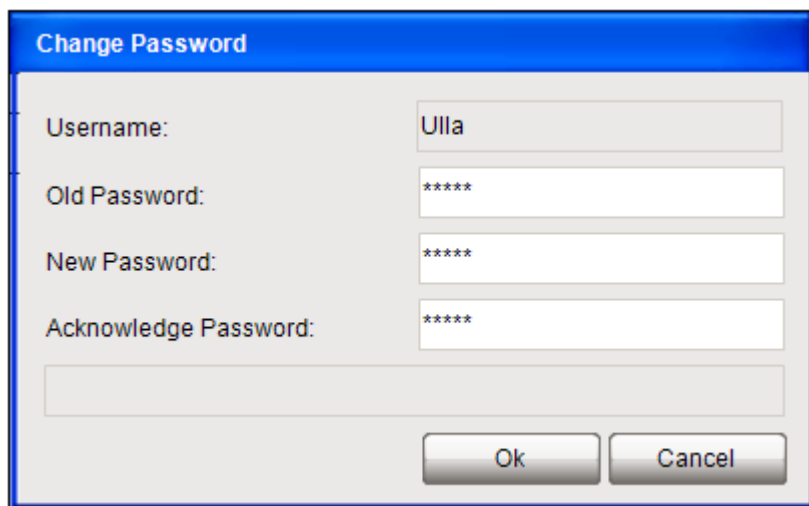
Dialog and actions:

- *Login*: dialog VUM_Login will open.
- *Logout*: a logout procedure will be run.
- *Change user password*: dialog VUM_ChangePassword will open.
- *Open user configuration*: dialog VUM_UserManagement will open.



The image shows a 'Login' dialog box with a blue title bar. It contains two text input fields: 'Username:' with the text 'Ulla' and 'Password:' with six asterisks. Below these fields is an empty text area. At the bottom right, there are two buttons: 'Ok' and 'Cancel'.

Figure 8-144. VUM_Login



The image shows a 'Change Password' dialog box with a blue title bar. It contains four text input fields: 'Username:' with the text 'Ulla', 'Old Password:' with six asterisks, 'New Password:' with six asterisks, and 'Acknowledge Password:' with six asterisks. Below these fields is an empty text area. At the bottom right, there are two buttons: 'Ok' and 'Cancel'.

Figure 8-145. VUM_ChangePassword

Username	Fullname
Ulla	Ulla Hoppla
Peter	Peter Smith
Paul	Paul Brown

Username: Admin

Fullname: Administrator

Password: *****

Acknowledge Password: *****

Usergroup: Admin

Deactivate:

Ok Cancel

Figure 8-146. VUM_UserManagement

Close Dialog

Here you can define that on the event the specified dialog will be closed with the specified result. Choose the desired dialog from the selection list, which offers all currently available **Input Dialogs**.

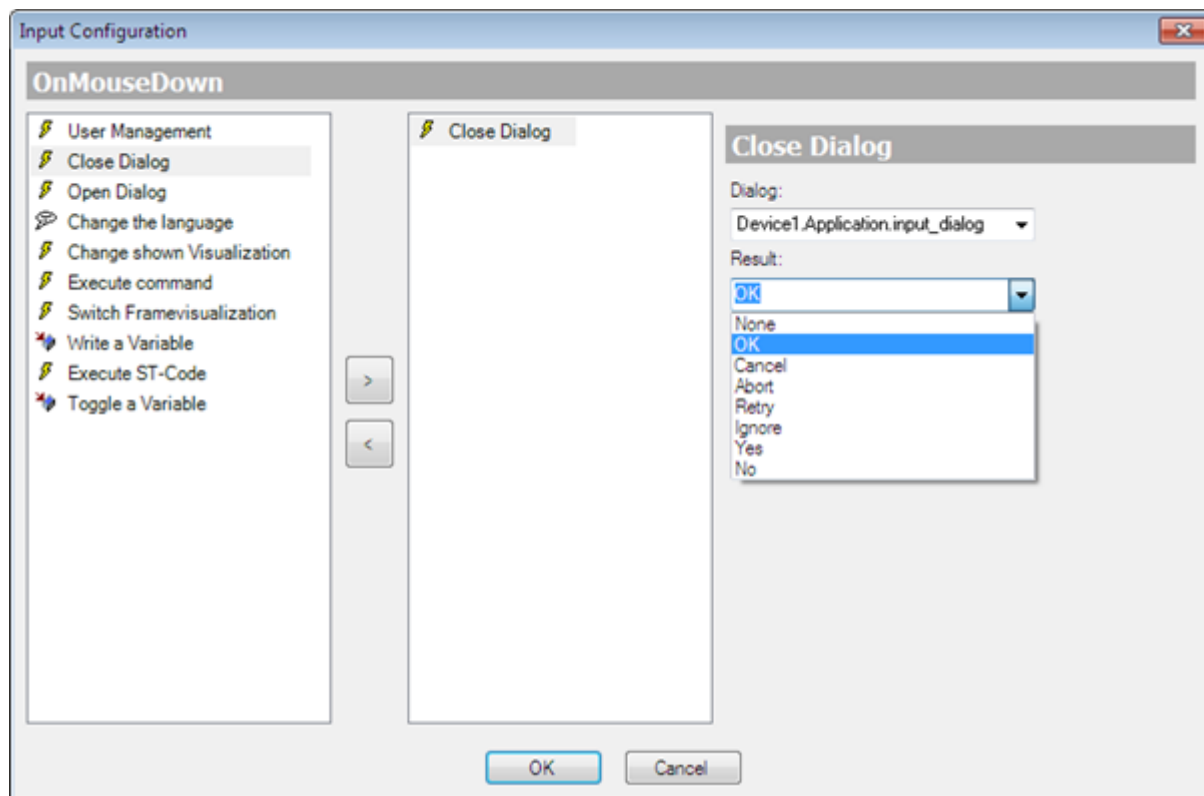


Figure 8-147. Input Configuration for OnMouseDown with selected Close Dialog

Dialog: select a dialog from the drop-down list.

Result: the result list offers the standard options which are used in dialogs that require a user response: OK, Cancel, Abort, Retry, Ignore, Yes, No.

Notice that the result, given by the lastly closed dialog, can be scanned and a respective reaction on this result can be configured in any element of the current visualization. For this purpose use the input configuration parameter *OnDialogClosed*.

See **Input Dialogs** for information on default and user-defined visualization input dialogs.

Open Dialog

Here you can configure a dialog opens on a mouse-action, which is realized by another (default or user-designed) visualization. The selection list will offer all visualizations which are defined in their **Properties of a Visualization Object**. as to be used as a dialog. So you might provide a self-designed dialog as user input mask in a visualization.

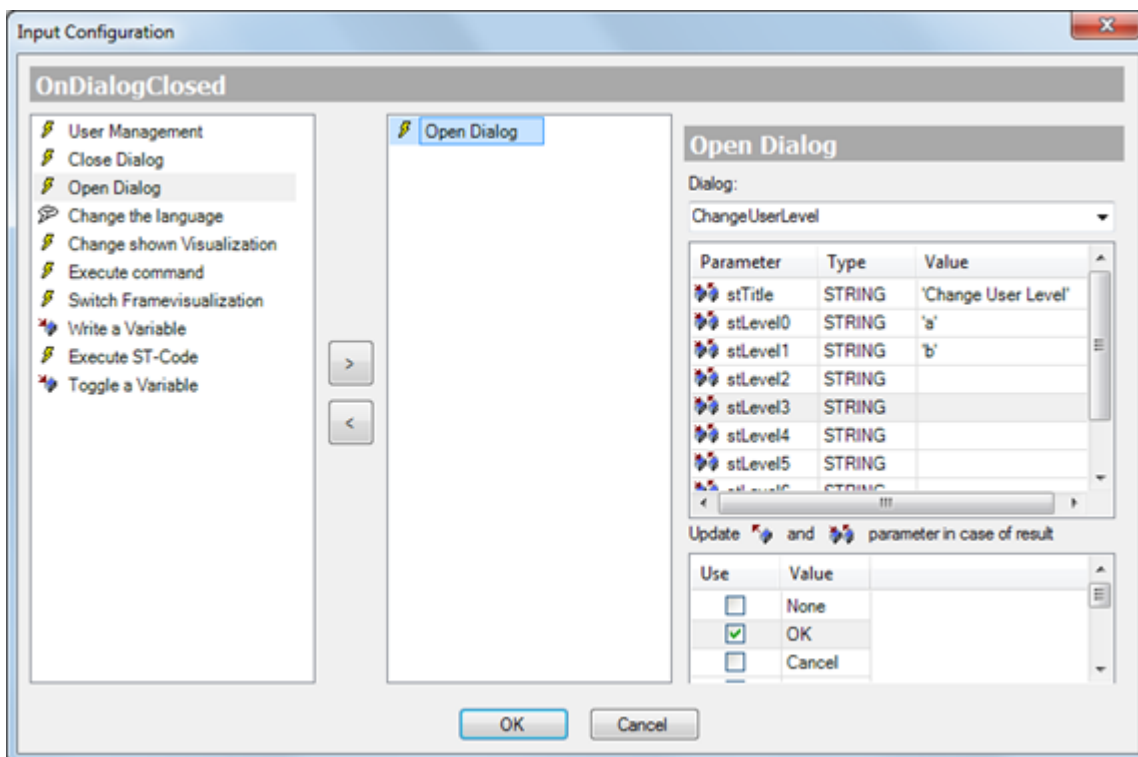


Figure 8-148. Input Configuration, with selected Open dialog

For the selected dialog visualization, the input (VAR_INPUT) and inout (VAR_IN_OUT) parameters as defined in the visualization interface will be displayed (Parameter, Type, Value). You can replace the default values (=0) by visualization-specific values, which then will be written to the dialog visualization each time the dialog gets opened.

- *Result*: set a check in the *Use* column of the desired result in *Valor*.
- *Valid Values*: Possible values are None, OK, Cancel, Abort, Retry, Ignore, Yes, No.

In difference to previous versions of the visualization, for a standard usage of a dialog visualization it is not any necessary to configure a *OnDialogClosed* action in order to get a dialog closed. However this action is still available as an option for special needs.

ATTENTION:

Note that the Output and InOut parameters of a dialog visualization will not be written before the dialog gets closed! Until then the values are just stored on the stack, i.e. they are not handled as references but as copies.

Change the Language

Enter the language which should be used for the display of the visualization texts. Use the language identifier as defined in the corresponding textlist(s). Alternatively you may click on for selecting the appropriate language within the input assistant.

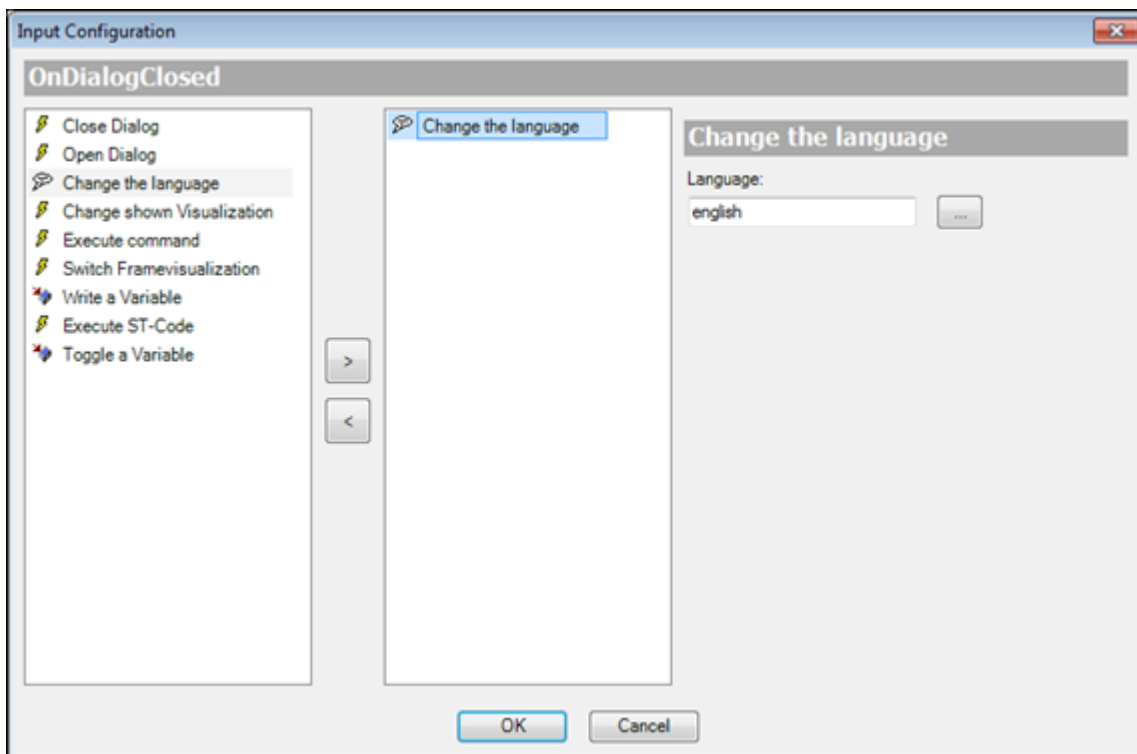


Figure 8-149. Input Configuration with selected Change the language

Change Shown Visualization

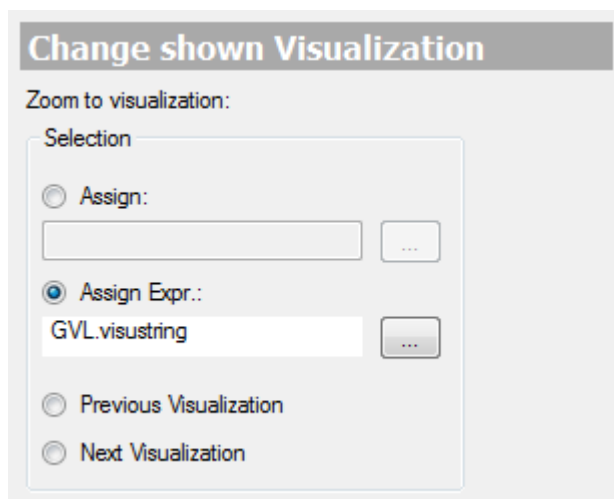



Figure 8-150. Input Configuration with selected Change Shown Visualization

Select one of the following options to define which visualization should be displayed in online mode on the respective mouse action.

Assign: A visualization can be specified. The selected visualization must be entered with the complete path, that is <device name>. <application name>.<visualization name>. For this purpose best use the input assistant via the . Example: MyPlc.App11.Visu_xy

Assign Expr.: A global variable can be specified which determines the name of the visualization to be shown. The Input Assistant will open for this purpose. The selected variable will be entered, for example *GVL1.visustring*.

The sequence in which visualizations get displayed in online mode due to user inputs will be stored internally. With the following two options you can jump back or forward within this sequence:

Previous Visualization: The previously displayed visualization will be shown again. If no visualization had been displayed previously, the current one will be kept.

Next Visualization: The visualization following to the active one in the stored sequence of display will be shown. This is only possible after previously having changed back from this visualization via Previous Visualization.

Execute Command

Execute programs, print, recipe manager commands.

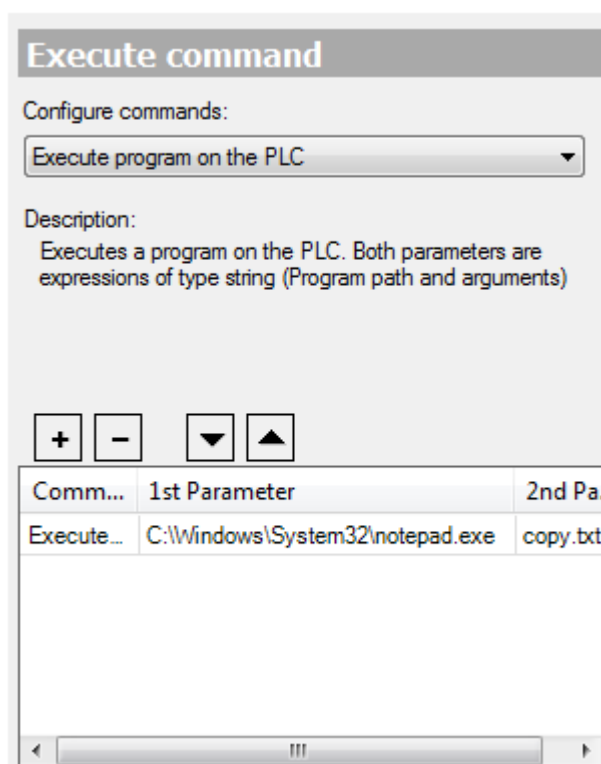


Figure 8-151. Input Configuration with selected Execute command

Here you can define a sequence of commands to be executed on the mouse action.

From selection list *Configure commands* choose the desired command and use button **+** in order to enter the command in the table in the lower part of the dialog. This table always contains the commands currently selected for the local **Input Configuration**.

A short description on the command currently selected in the upper list always appears in the center part of the dialog. Accordingly supply the parameters for the command in the table columns *1st Parameter* and *2nd Parameter*. (Column *Command* shows the internal command name.) See also further below here on the help page the descriptions of the particular commands.

Via button **-** the currently selected entry can be removed from the table. Later in online mode, when the user input will be performed on the visualization element, the configured commands will be

executed in that order in which they are arranged here in the table from up to down. You can change this order by shifting a currently selected entry by the buttons ▼ or ▲

The following commands are available:

Command	Description
Execute program on the PLC Execute program on the client	Executes the given program (*.exe) on the controller (PLC) resp. the visualization client. 1st parameter: STRING, path of the program (for example "C:\programs\notepad.exe") 2nd parameter: STRING, arguments needed by the program (for example name of file to be opened: "copyfile.txt")
Print	The standard Print dialog gets opened, where settings for the print area, the printer parameters etc. can be configured and a printout of the current visualization can be generated. Please note that this command is only supported on target visualizations on Windows operating systems

Table 8-44. Comandos de Execução

ATTENTION:

At **Input Configuration** the Execute Command action can't be used in the web visualization, with 'Execute program on client' command due to security reasons.

For the following commands concerning recipes, notice the following:

The *recipe definition* is an object in the *device tree* below the *Recipe Manager*. It consists of a list of variables and the recipes (that is a group of variables values) defined for these variables. It can be edited in the Recipe Manager editor window.

The *recipe name* is the name of a particular recipe for the variables within a recipe definition. The recipe names are there visible as column headers, see Figure 8-152:

1. Recipe Definition Name
2. Recipe Name

Variable	Type	Name	Comment	Minimal Value	Maximal Value	Current Value	R1	R2
MainPrg.bvar	BOOL	bv					FALSE	TRUE
MainPrg.ivar	INT	iv	Degree...	0	30		33	20
MainPrg.visutext	STRING	sv					'ele_1'	'ele_2'

Figure 8-152. Recipe Definition, Recipe name

Command	Description
Create recipe	A new recipe will be created in the specified recipe definition. 1st parameter: Recipe definition name 2nd parameter: Name for new recipe
Read recipe	The current values of the variables of the specified recipe definition will be read from the controller and be "written" to the specified recipe. This means that the values will be stored implicitly (in a file on the PLC) and also immediately will be monitored in the recipe definition table in the Recipe Manager. In other words: The recipe managed in the Recipe Manager gets updated with the actual values from the PLC.

	<p>1st parameter: Recipe definition name</p> <p>2nd parameter: Name of recipe for which the values should be read from the PLC and be stored as actual recipe values.</p>
Write recipe	<p>The values of the given recipe, as visible in the Recipe Manager, will be written to the variables on the PLC.</p> <p>1st parameter: Recipe definition name</p> <p>2nd parameter: Name of the particular recipe the values of which should be written to the variables on the PLC.</p>
Save a recipe in a file	<p>The values of the specified recipe will be written to a file with extension *.txtrecipe, the name of which you have to define. For this purpose the standard dialog for saving a file in the local file system will open.</p> <p>Attention: The implicitly used recipe files, necessary as a buffer for reading and writing of the recipe values, may not get overwritten. This means that the name for the new recipe file must be different from "<recipe name>.<recipe definition name>.txtrecipe" !</p> <p>1st parameter: Recipe definition name</p> <p>2nd parameter: Name of the particular recipe to be saved in the specified file</p>
Load a recipe from a file	<p>The recipe which has been stored in a file (see above "Save recipe") can be reloaded from this file. The standard dialog for browsing for a file will open for this purpose. The filter automatically is set to extension "*.txtrecipe". After having reloaded the file, the recipe values will be updated accordingly in the Recipe Manager.</p> <p>1st parameter: Recipe definition name</p> <p>2nd parameter: Name of the particular recipe, for which the values should be loaded from the specified file to the Recipe Manager.</p>
Delete recipe	<p>The specified recipe will be removed from the specified recipe definition.</p> <p>1st parameter: Recipe definition name</p> <p>2nd parameter: Recipe name</p>

Table 8-45. Comandos de Recipe

Each mouse-input which has got assigned any action via the **Input Configuration** dialog, will get the property *Configured* in the Properties table.

Switch Framevisualization

Preconditions: There is a visualization (or several) in your project, containing frame elements, which have got assigned several visualizations via the **Frame Selection**. command (*Frame Configuration* dialog). These visualizations get numbered within the frame by indices 0, 1, 2, etc. and by default in online mode the first of the assigned visualizations (index 0) will be displayed in the frame.

Switch Framevisualization: In the current configuration dialog you can define, that - if the chosen mouse action is performed on the current visualization element in online mode - in a specified frame a definite one of its assigned visualizations should be displayed. So you can use the current visualization element as a switch for the display of the available visualizations in a frame..

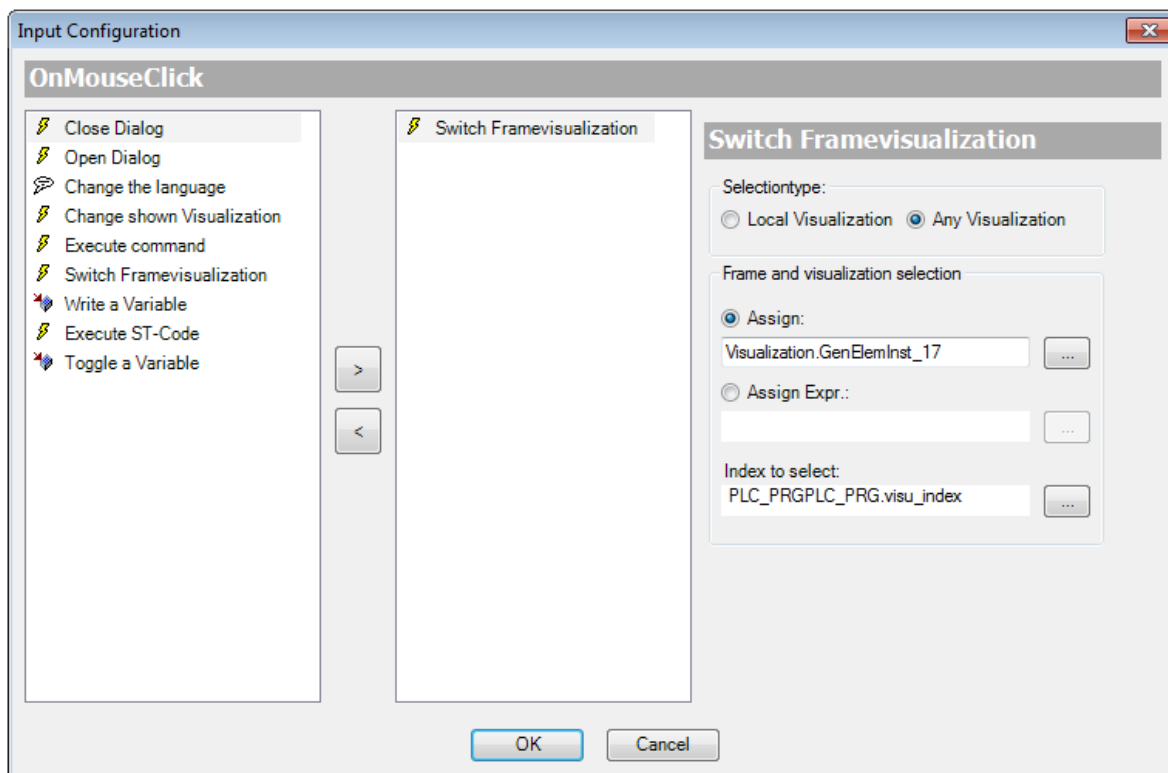


Figure 8-153. Input Configuration with selected Switch Framevisualization

Selection type: The selection of the frame element:

- can be limited to frames which are inserted in the current (local) visualization. This means getting a simplified configuration dialog without the possibility of dynamic configuration > *Local Visualization*.
- can be extended to all visualizations available in the project, which also allows dynamic definitions of the frame and the visualization to be displayed > *Any Visualization*.

Local Visualization: Only frames of the currently edited visualization can be addressed and thus are available in the Frame selection field. This is a simple dialog for quick and direct configuration within the local visualization, however does not provide the possibility to define the desired visualization dynamically via a project variable. If dynamic definition is required, use option *Any Visualization*, see below.

Frame selection shows the locally available frames and each indented below the assigned visualizations.

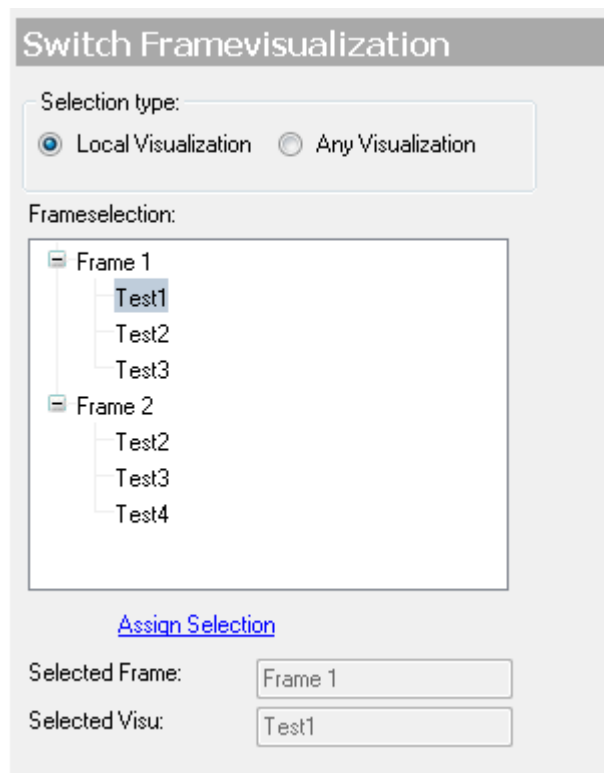


Figure 8-154. Frame selection, local

Below a frame entry select that visualization which should be displayed on the respective mouse action in online mode. Click on *Assign Selection* in order to store the selection. The current setting then will be displayed in the fields Selected Frame and Selected Visu.

Any Visualization: All frames + assigned visualizations, which are available in the whole project can be addressed in the current configuration. In this case the frame and the visualization also can be defined by a project variables, that is dynamically. The 'Frame and visualization selection' will be done via the following fields and options:

The desired frame element can be defined directly (Assign) or by a variable (Assign Expr.):


Assign: Direct assignment; if this option is activated, the complete path of the frame element must be entered. Via button  the input assistant can be used. Syntax: <device name>.<application name>.<visualization name>.<frame element name>. See an example in the following image:

Figure 8-155. Seleção de Frame, atribuição direta

Assign Expr.: This option can be used alternatively, if a project variable should dynamically define the frame element. The variable must be of type `STRING`. Via button the input assistant can be used. The variable must specify the complete path of the frame element. Syntax: <device name>.<application name>.<visualization name>.<frame element name>. Example: `frame_var: STRING:='Test1.Frame1'`;

The desired visualization to be displayed on the mouse action in the above defined frame, is to be specified by its index number. This index results from an integer numbering of all visualizations assigned to a frame in the 'Frame Configuration' dialog. The index number of the first visualization in this list is "0". So initially by default this first visualization will be displayed in online mode.

Index to select: Enter here the index number of the desired visualization, either directly (for example "1") or via a variable used by the application. Via button , para selecionar uma variável.

Write a Variable

Edit field, numpad, keypad.

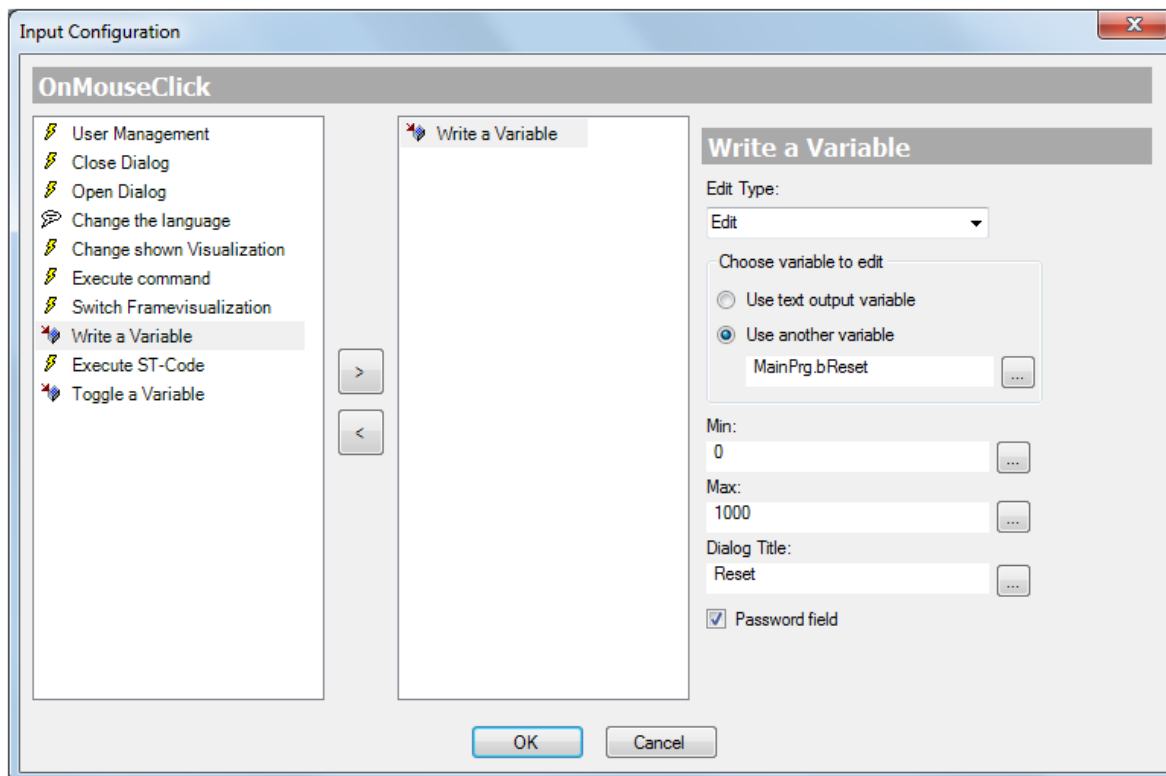


Figure 8-156. Input Configuration with selected Write a variable

Se existe uma **Input Configuration** of type *Write a Variable* for a visualization element, then as soon as the respective mouse action is performed on this element in online mode, the element will provide a possibility to enter a value. The value can be typed in as a sequence of characters or via a numpad or keypad and after having terminated the input will be written to the project variable specified in the Input Configuration dialog. The value will be interpreted according to the type of the project variable as a text string or as a numeric value.

Using the Configuration Dialog

From the selection list *Edit Type* choose one of the following input types:

- *Edit*: The mouse action will open an edit field, where you can type in a string or numeric value.
- *VisuDialogs.Keypad*: The mouse action will open a simulated keypad and a text string can be entered by mouse-clicks on the respective keys.
- *VisuDialogs.Numpad*: The mouse action will open a simulated numpad and a numeric value can be entered by mouse-clicks on the respective keys.
- *<visualization name>*: Name(s) of available visualization(s) which is/are defined in **Properties of a Visualization Object** to be a *numpad/keypad* visualization. If one of these is set as input type, then the mouse action will open the respective, visualized numpad or keypad.

Choose variable to edit: Specify here the variable to be written with the value typed in by the user in online mode:

- *Use the text output variable*: The value will be written to the variable specified in the variables for text variables of the visualization element.

- *Use another variable*: Enter any project variable. The input assistant might be opened for this purpose via the button.

A minimum and maximum can be defined for the value to be written to the specified variable. Enter directly a *Min.* and *Max.* value or a project variable defining the resp. limit.

A title which should be displayed in the dialog title bar can be defined via a text string or a text variable in the *Dialog title* field.

If you want the entry to be displayed hidden by ******* (as in case of a password), mark the checkbox in front of *Password field*.

Example

See for example the Figure 8-156 and assume that this is the configuration dialog of a rectangle element. If in online mode the mouse-key is pressed on this rectangle, a numpad will be displayed, showing the title given by `MainPrg.MyTitle`.

Enter a value: For example by mouse-clicks on 1, 2 and 3 enter 123. This value will be displayed in the upper part of the dialog, where also the minimum and maximum values as defined by `MainPrg.minVal` and `MainPrg.maxVal` are shown. As soon as you have confirmed the input by a mouse-click on the numpad OK button, 123 will be written to variable `MainPrg.ivar`. If `ivar` is declared of type `STRING`, it will get value 123, if it is of a numeric type it will get value 123.

Execute ST-Code

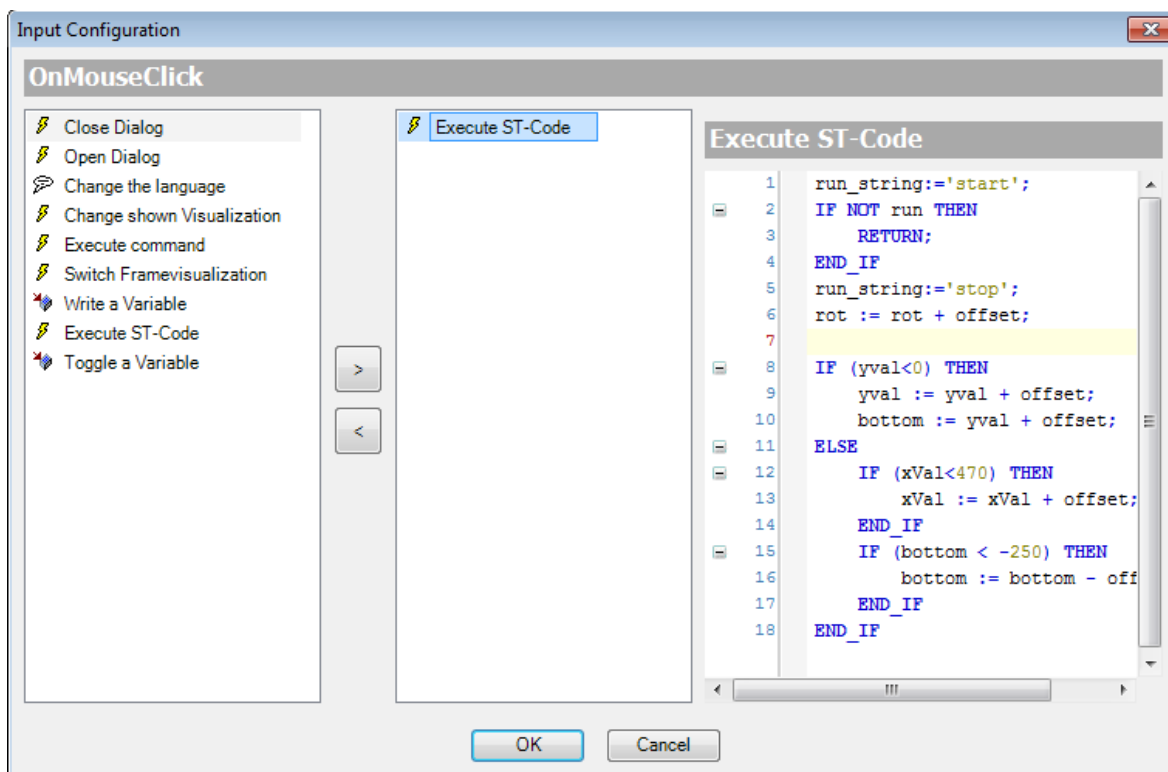


Figure 8-157. Input Configuration with selected Execute ST-Code

In the editor field enter the code in Structured Text (ST) which should be executed on the mouse action in online mode.

Toggle a Variable

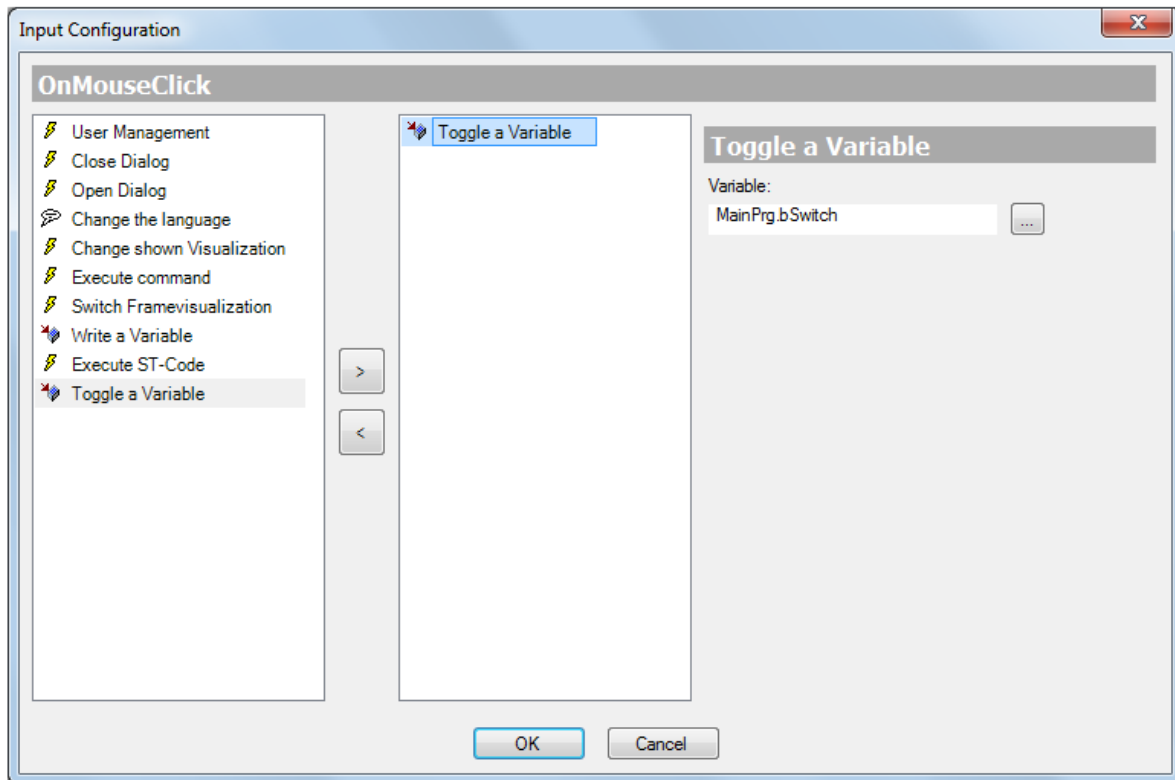


Figure 8-158. Input Configuration with selected Toggle a Variable

Enter a Boolean project variable, the value of which should be toggled between TRUE and FALSE by the mouse action. Example: MainPrg.bSwitch With click on the *Input Assistant* will help .

Access Rights

A click in the value field of *Access Rights* in the *Properties* view of an element opens this dialog.

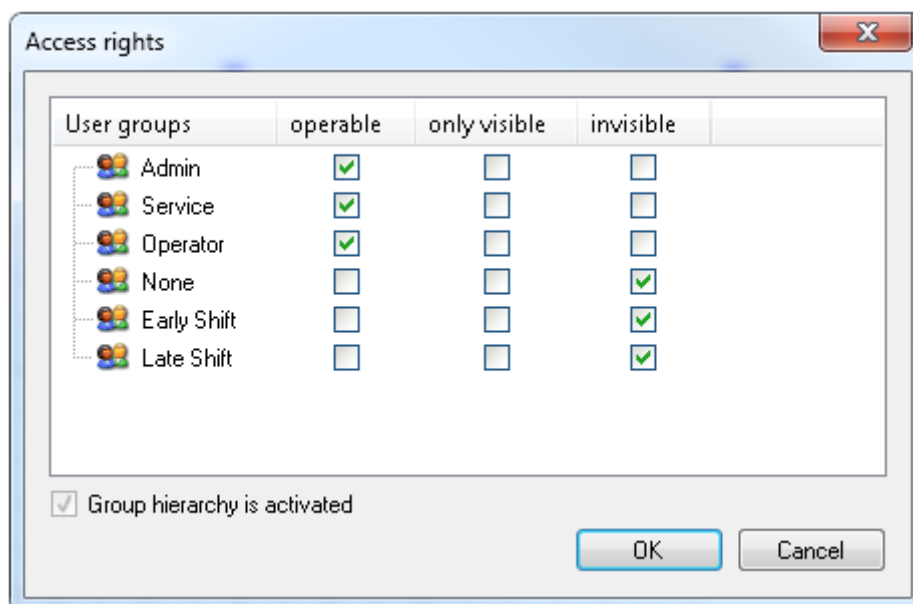


Figure 8-159. Access Rights

User groups: the groups configured in *VisualizationManager > User management > Groups* are listed here. Mark a checkbox in the row of the group to assign access rights to them.

Operable: if the checkbox is marked, the element offers full functionality.

Only visible: the element offers no functionality but is displayed.

Invisible: the element is not displayed.

Group hierarchy is activated: if group hierarchy set to use in *VisualizationManager > User management > Settings*, this is displayed here and a tick is set in the deactivated checkbox.

Note:

Default group None: *None* is the default group and cannot be deleted. If no user is logged in, the visual elements behave as configured for it.

When limiting the access rights of an element, it is recommended to specify *None* with the least access rights.

Properties

General

Property	Value Description
Elementname	The name of the element can be changed. Default name is GenElemInst_x, x represents consecutive numbering
Type of element	Shows the Element Type

Table 8-46. Properties General

Position

The element is defined with position (X/Y-coordinates) and size (width and height), all in pixels. The origin is the upper left window border. The positive X-axis points to the right and the positive Y-axis points down. With editing the value fields, the displayed element in the editor is changed simultaneously.

Property	Value Description
X	Defines the horizontal position. X=0 is the left window border. The positive axis points to the right.
Y	Defines the vertical position. Y=0 is the upper window border. The positive axis points down.
Width	Horizontal length of the element.
Height	Vertical length of the element.

Table 8-47. Properties Position


Center


With editing the value fields, the associated element  in the editor is moved simultaneously.

Property	Value Description
X	X-position of the turning point of the element.
Y	Y-position of the turning point of the element

Table 8-48. Properties Center

Colors

A color is defined by a hex number which is composed of the Red/Green/Blue (RGB) components. For each color value 256 (0-255) colors are available. Here the settings are static. The color can be chosen from a selection list or in the color selection dialog to be opened via button . Example:

 0; 128; 64

Property	Value Description
Normal state: - Frame color - Fill color	Select a frame and fill color for the element in normal state. In case of the variable defined in Color variables > Toggle color is FALSE, the element is in normal state.
Alarm state: - Frame color - Fill color	Select a frame and fill color for the element in alarm state. In case of the variable defined in Color variables > Toggle color is TRUE, the element is in alarm state.

Table 8-49. Properties Colors

Element Look

Property	Value Description
Line width	Width of the outline in pixels. 0 encodes as 1 and set the line width to 1 pixel. If you want to have no frame you have to set "Line style" to hollow.
Fill attributes	Select one of these attributes applied to "Fill color" - Filled: the fill color is visible. - Hollow: the fill color is invisible.
Line style	Selecione um dos seguintes tipos de estilo aplicado ao contorno: - Solid - Dashes - Dots - Dash Dot - Dash Dot Dot - Hollow: then the outline is invisible.

Table 8-50. Properties Element Look

Texts



Property	Value Description
Text	The entered text labels the element. A text format string is also possible.
Tooltip	The entered text appears as element tooltip when the visualization is in online mode.
Horizontal alignment	Define the horizontal alignment of the text by selecting one of: - Left - Centered - Right
Vertical alignment	Define the vertical alignment of the text by selecting one of: - Top - Centered - Bottom
Font	Select on of the predefined fonts: - Default - Headline - Large - Title - Annotation With click on  , a dialog for a user-defined setting of the font properties will open.
Font Color	Select a font color for the element from the drop-down list or set one in the dialog which opens when  is clicked.

Table 8-51. Properties Texts

Absolut Movement

The element can be moved by changing the X- and Y-position (pixels) of the upper left corner of the element by a variable of type INT. Here, absolute coordinates are used.

Property	Value Description
Movement - X - Y	X: Enter an integer variable specifying the current X- offset of the upper left corner of the element in pixels for shifting the element in X-direction (positive value shifts the element from left to right). Y: Enter an integer variable specifying the current Y-offset of the upper left corner of the element in pixels for shifting the element in Y-direction (positive value shifts the element from top to bottom).
Rotation	Enter an integer variable defining the angle (angular degrees) by which the element will be moved around the turning point Center \oplus . positive values = clockwise Notice that the element itself will not rotate, in contrast to Interior rotation.
Scaling	Enter an integer variable defining the current scale value (percent); the element size will be changed linear according to this value. Implicitly the value will be divided by 1000, so that it is not necessary to use REAL-variables in order to get a reduction of the element. The scaling always will refer to the turning point: Center \oplus .
Interior rotation	Enter an integer variable defining the angle (angular degrees) by which the element will rotate around its turning point: Center \oplus . positive values = clockwise. Notice that the element itself will rotate, in contrast to Rotation

Table 8-52. Properties Absolut Movement

Relative Movement

The element can be shifted relative to its fix position defined in *Position*. The top-left and bottom-right edge of the element will be moved due to the values getting in the integer variables in X/Y-direction. This specifies in contrast to the absolute movement a relative position being the distance to the initial position. Because of that the element can change its form.

See *Absolute movement* for shifting the element as a whole.

Property	Value Description
Movement top-left - X - Y	X: integer variable, which value set the number of pixels by which the top left edge will be shifted in X-direction. Y: integer variable, which value set the number of pixels by which the top left edge will be shifted in Y-direction.
Movement bottom-right - X - Y	X: integer variable, which value set the number of pixels by which the bottom right edge will be shifted in X-direction. Y: integer variable, which value set the number of pixels by which the bottom right edge will be shifted in Y-direction.

Table 8-53. Properties Relative Movement

Text Variables

These are dynamic definitions, see also *Dynamic texts* for using text lists. For static definitions see *Texts*.

Property	Value Description
Text variable	The assigned variable defines the element text (text output variable).
Tooltip variable	The assigned STRING variable defines the tooltip text of the element.

Table 8-54. Properties Text Variables

Dynamic Texts

These parameters serve for a dynamic text definitions basing on text lists (allows language switching). For details on text lists see under chapter **Menu Text List**. For another dynamic text definition by providing the text via a string variable see category *Text variables*. For static text definitions see category *Text*.

Property	Value Description
Text list	Name of the textlist object (string) as used in the POUs or devices view. For example errorlist.
Text index	Index of the text as defined in the textlist (string). Enter the ID directly, for example 'M1' or a string variable for example error_22.
Tooltip index	Index of the text as defined in the textlist (string). Enter the ID directly, for example 'T1' or a string variable for example error_tooltip_22.

Table 8-55. Properties Dynamic Texts

Font Variables

These are used for dynamic font definitions of the element text via project variables. Static definitions are done in *Text properties > Font*.

Property	Value Description
Font name	Enter a variable of type STRING providing the name of the font to be used for the label of the element (naming as used in the standard dialog for defining the font), for example plc_prg.font_var (font_var := Arial;)
Size	Enter a variable of type INT providing the size of the element text in pixels (as used in the standard dialog for defining the font) for example prog1.font_height (font_height:=16;).
Flags	Enter a variable of type DWORD defining the font display by one of the flags listed below. A combined definition can be reached by addition of the flag numbers: 1: italic 2: bold 4: underlined 8: canceled For example prog2.font_type: if font_type:=6, the text will be displayed bold and underlined.
Charset	The character set for the font to be used can be specified by the standard character set number. Enter an variable of type DWORD providing this number (see also the "Script" selection in the default font selection dialog)
Color	Enter a DWORD variable specifying the color for the element text.

Table 8-56. Properties Font Variables

Color Variables

Applied for dynamic definitions for the color of the element via a project variable.

A color is defined by a hex number which is composed of Red/Green/Blue (RGB) components. A project variable of type DWORD must be entered like prog1.dwFillColor.

The variable must define the color value in hexadecimal format. To fill the complete DWORD, the first two digits after "16#" have to be set to zero: 16#00RRGGBB.

Color Value

```
dwFillColor := 16#008FE03F;
```

Hex numbers:

- 8F: red
- E0: green
- 3F: blue

Property	Value Description
Toggle color	Enter a Boolean variable which toggles the element color between "Normal state" (variable = FALSE) and "Alarm state" (variable = TRUE).
Normal state: - Frame color - Fill color	Enter a DWORD variable determining the fill and frame color in normal state of the element. This overwrites the values currently set by Colors > Normal state . The values of the project variables are used when the variable defined in "Toggle Color" is FALSE.
Alarm state: - Frame color - Fill color	Enter a DWORD variable determining the fill and frame color in alarm state of the element. This overwrites the value currently set by Colors > Alarm state . "Frame Color" and "Fill Color" are used when the variable defined in "Toggle Color" is TRUE.

Table 8-57. Properties Color Variables

Look Variables

Applied for a dynamic definitions for the look of the of the outline and of the filling of the element. Static definition have to be set in "Element look".

Property	Value Description
Line width	Enter a integer variable defining the line width of the element in pixels. This overwrites the value currently set by Element look > Line width.
Fill attributes	Enter a DWORD variable defining the fill attributes of the element. The color, as defined by the color variables, can be displayed or ignored: Variable value = 0: filled. Variable value > 0 : hollow means the filling is invisible. This overwrites the value currently set by Element look > Fill attributes .
Line style	Enter a DWORD variable defining the style of the element outline. Following values are equal to following line style: 0: Solid 1: Dashes 2: Dots 3: Dash Dot 4: Dash Dot Dot 8: Hollow: outline is invisible

Table 8-58. Properties Look Variables

State Variables

Property	Value Description
Invisible	Enter a Boolean variable; if the variable gets TRUE, the element will be invisible in online mode.
Deactivate inputs	Enter a Boolean variable; if the variable gets TRUE, any inputs on the element will be of none effect.

Table 8-59. Properties State Variables

Access Rights

Property	Value Description
Access rights	<p>With click, Access rights opens.</p> <p>Status messages:</p> <ul style="list-style-type: none"> - Not set. Full rights: this default message is set if the element is displayed operably for all groups. - Rights are set: Limited rights: this message is set if the element is displayed limitedly for at least one group.

Table 8-60. Access Rights

Input Configuration

Here, you define what after-action should happen if the user in online mode performs an input on an element. The input events an element can have are:

- *OnDialogClosed*: This event is given by closing of any dialog within the visualization, which has been opened before for user input purposes.
- *OnMouseClicked*: This mouse event is given when the cursor is pointing on the element and full a mouse-click (press and leave the mouse-button) is performed.
- *OnMouseDown*: This mouse event is given when the mouse-button is pressed when the cursor is pointing on the element (press the mouse-button).
- *OnMouseEnter*: This mouse event is given when the cursor is being drawn on the element.
- *OnMouseLeave*: This mouse event is given when the cursor leaves the element.
- *OnMouseMove*: This mouse event is given when the cursor is being moved within the element.
- *OnMouseUp*: This mouse event is given when the mouse-button is released on the element. The click down of the mouse-button is done before outside the element.

Click *Configure...* for opening the dialog **Input Configuration** to assign the after-actions. Click in the field beside the input you want to configure. For every selected after-action a new entry in the property table is configured. Every input can get a arbitrarily assigned number of after-actions.

Tap

With *Tap* you can configure, that when the event Tap occurs, the value of a project variable is set depending on the mouse behavior.

Property	Value Description
Variable	Here you can enter a Boolean variable, which is TRUE if the mouse-button is pressed while the cursor is pointing on the element and which gets FALSE again if the mouse-button is released.
Tap FALSE	If this option is activated, the tap-behavior described for the above define variable will be reversed. That means that the variable will be FALSE as soon as the mouse-button is pressed and it will get TRUE when releasing the button.
Tap on enter if captured	If this option is activated, the variable value reacts within the element as described under "Variable" as long as the mouse is in the element area. Additionally it reacts on a drag of the mouse. If the element area is left while the mouse button is pressed, the variable gets FALSE. But it gets TRUE again, if the mouse is dragged back into the element area. Therefore it is taken into account, that the mouse is captured as long as the mouse button is pressed even if the element area is left

Table 8-61. Input Configuration Tap

Toggle

With *Toggle*, you can enter an Boolean variable, which alternately will get TRUE or FALSE on each mouse-click on the element.


Property	Value Description
Variable	Enter a Boolean variable name. The input assistant can be used via button  . The variable value toggles with every mouse-click on the element between TRUE and FALSE. If the cursor is moved out of the element while the mouse key is pressed, the variable value is not toggled. So a started toggle-input can be aborted.
Toggle on up if captured	If this option is activated, the variable value reacts within the element as described under "Variable" as long as the mouse is in the element area. Additionally it reacts on a drag of the mouse. If the element area is left while the mouse button is pressed, the variable is toggled. Therefore it is taken into account, that the mouse is captured as long as the mouse button is pressed even if the element area is left.

Table 8-62. Input Configuration Toggle

Hotkey

With *Hotkey* you can define a key perhaps in combination with an modifier key and associate a certain event (MouseDown/MouseUp), which should be executed on the respective event of the key (KeyDown/KeyUp). Per default on KeyDown, the MouseDown action gets executed and on KeyUp the MouseUp action. This might be useful if a visualization should be operated via mouse as well as via keyboard, because the input actions must be configured only once. This part of the element configuration will also be administered in the **Hotkeys Configuration** of the visualization. Any changes always will be synchronized immediately in the hotkeys editor as well as in the element properties.

Property	Value Description
Key	Assigne a key. A selection list provides all currently supported keys, for example [M] . Hotkeys Configuration.
Events	Defines the events to be executed when using the key. Possible values provided in a selection list: - None: no action - MouseDown: OnMouseDown-action when key is pressed - MouseUp: OnMouseUp-action when key is released - MouseDown/MouseUp: OnMouseUp and OnMouseDown actions when key is pressed or released
Shift	If this option is activated, the key must be combined with [Shift] .
Control	If this option is activated, the key must be combined with [Ctrl] .
Alt	If this option is activated, the key must be combined with [Alt] .

Table 8-63. Input Configuration Hotkey

Basic

The Basic element group is compound by Rectangle, Round Rectangle, Ellipse, Line, Polygon, Polyline, Bézier Curve, Pie, Image and Frame.

Editing Elements

User Inputs when Editing an Element

For editing an element in the visualization editor following user inputs are provided:

- a set of visualization commands affecting at least one element: **Menu Visualization or Visualization Commands.**
- different mouse operations which can be done on an element: **User Inputs When Editing an Element.**


User Interface when Editing the Properties

Properties define the characteristic of a visualization element:

- its appearance: changes on properties responsible for the appearance of the element are immediately visible in the visualization editor.
- its arrangement in the visualization.
- its animation: the visualization element can be animated by direct use of project variables.
- actions to be executed after an user input on the element.

A property can be modified by editing the *Value* field. For this purpose depending on the property type, you can open an edit frame, a selection list, a dialog, or activate a checkbox. Such a widget will open.

- by a double-click,
- by a single click in a selected field,
- or via [Space] bar when the field is already selected.

If a variable has to be assigned, use  for opening the *Input Assistant*. Category *Variables* lists all variables defined in the project up to know.

Rectangle, Round Rectangle, Ellipse

Rectangle, round rectangle and ellipse are of the same element type. They can be converted to each other just by changing the property *Type of element*.

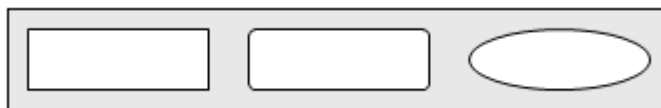


Figure 8-160. Rectangle, Round Rectangle and Ellipse

Properties of Rectangle, Round Rectangle, Ellipse

The elements *Rectangle*, *Round Rectangle* e *Ellipse* have no special features in the properties settings, their settings are identical to those presented in section Properties.

Line

Este element é uma linha simples.

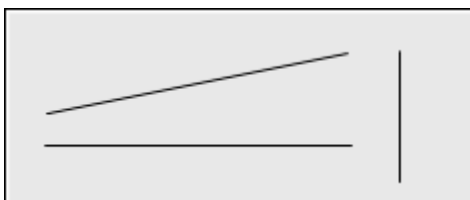


Figure 8-161. Line

Properties of Line

The element *Line* has all configuration settings presented in section Properties, but with some special features, which are listed in sequence:

Position: for this element the position is given by the X and Y coordinates of the points that form such elements.

Colors: this element has options a little different from those in the Table 8-49:

- *Color:* Select a base color for the element filled with when the element is in normal state. In case of the variable defined in *Color variables > Toggle color* is FALSE, the element is in normal state.
- *Alarm Color:* Select a color, the element is filled with when the element goes in alarm state in case of the variable defined in *Color variables > Toggle color* is TRUE.

Element Look: this element does not have the configuration option *Fill attributes*.

Relative Movement: the relative movement of this element is defined for each of their points.

Color Variables: this element has options a little different:

- *Color:* Enter a DWORD variable determining the color of the element. This overwrites the value currently set by *Colors > color*. *Color* is used when the variable defined in *Toggle Color* is FALSE.
- *Alarm Color:* Enter a DWORD variable determining the alarm color of the element. This overwrites the value currently set by *Colors > color*. *Color* is used when the variable defined in *Toggle Color* is TRUE.


Look Variables: this element has options a little different:

- *Line width variable Integral value:* Enter an integer variable defining the line width of the element in pixels. This overwrites the value currently set by *Element look > Line width*. Note that 0 encodes the same as 1.
- *Line style variable Integral value:* Enter an integer variable which provides the line style. Supported values are the same of the parameter *Line style*. This overwrites the fix setting in *Element look > line style*.

Polygon, Polyline, Bézier Curve

Polygon, polyline e Bézier curve are of the same element type. They can be converted to each other just by changing the property *Type of element*.

Element specific mouse input:

- Such an element can be defined with any number of points determine the section and the shape of the element. Default are five points: point [0] to [4]. Additional, points can be inserted in the editor: if the element is selected and the focus is on a point (small rectangle on frame line), the cursor shape is . Keeping pressed [Ctrl] while clicking on the focused point, another point is generated. A point can be removed by keeping pressed [Shift] + [Ctrl] and clicking on this point. Simultaneously the changes are regarded in *Properties* and *Position > Points*.

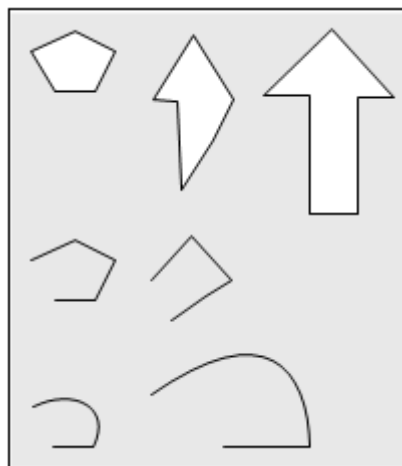


Figure 8-162. Polygon, Polyline, Bézier curve

Properties of Polygon, Polyline, Bézier Curve

The elements Polygon, Polyline, Bézier Curve have all configuration settings presented in section Properties, but with some special features, which are listed in sequence:

Position: for this element the position is given by the X and Y coordinates of the points that form such elements.

Relative Movement: setting option nonexistent.

Dynamic Points

Dynamic definition for the points defining the element.

Property	Value Description
Array of points	<p>Enter a variable which is a pointer to an array of structure <code>VisuElems.VisuStructPoint</code>. <code>VisuElems.VisuStructPoint</code> components <code>iX</code> and <code>iY</code> describes the X/Y-coordinates of a points. The array describes all points of the element and the structure can be filled dynamically by a project variable.</p> <p>The number of the elements points must be defined explicitly in "Number of points".</p> <p>Example: <code>pPoints : POINTER TO ARRAY[0..100] OF VisuElems.VisuStructPoint;</code></p>
Number of points	<p>Enter a variable of the type <code>INT</code> which defines the number of points of the element.</p> <p>Example: <code>iCount : INT:=24;</code></p> <p>The element is defined to have 24 particular points. This specification is needed because the definition of the particular points is given by the pointer, see "Array of points", that does not allow control on the number.</p>

Table 8-64. Properties Relative Movement

Pie

The default *pie* element inserted in the visualization editor is a full circle with the center position at the point the element was dropped.

Element specific mouse input:

- Start and end point of the pie can be moved by drag and drop of the little squares on the circular line in the editor. The associated properties *Start point* and *End point* are updated simultaneously.



Figure 8-163. Pie

Properties of Pie

The element Pie have all configuration settings presented in section Properties, but with some special features, which are listed in sequence:

Position: This element further has the following options:

- *Begin*: Defines the begin of the circle sector line.
- *End*: Defines the end of the circle sector line.
- *Variable for begin*: Enter here the desired variable name if you want *Begin* get defined dynamically by variable. For help from the *Input Assistant* click .
- *Variable for end*: Enter here the desired variable name perhaps with help of the *Input Assistant* if you want *End* get defined dynamically by variable.
- *Only show circle line*: Para ocultar as linhas de raio na Visualization do element Pie, habilite esta propriedade na caixa de seleção.

Relative Movement: setting option nonexistent.

Element Look: The *Line width* and *Line style* settings have no effect on Windows CE based systems.

Image

An image can be inserted in a visualization by this element. You assign an image to the element by specifying *Static ID* and enter the name of the image file. For unique access, extend the name by the name of an image pool. This can also be configured dynamically via a project variable containing the image ID for dynamic switching of images. For this, use *Bitmap ID*.

Directories providing images for the usage in a visualization are specified in the **Project Settings**, category *Visualization*.

A background image can be specified for the complete visualization by command **Background**.

Properties of Image

The element Image have all configuration settings presented in section Properties, but with some special features, which are listed in sequence:

Colors: this element has options a little different from those in the Table 8-49:

- *Color*: Select a base color for the element filled with when the element is in normal state. In case of the variabe defined in *Color variables* > *Toggle color* is FALSE, the element is in normal state.
- *Alarm Color*: Select a color, the element is filled with when the element goes in alarm state in case of the variabe defined in *Color variables* > *Toggle color* is TRUE.

Element Look: this element does not have the configuration option *Fill attributes*.

Color Variables: this element has options a little different:

- *Color*: Enter a DWORD variable determining the color of the element. This overwrites the value currently set by *Colors* > *color*. *Color* is used when the variable defined in *Toggle Color* is FALSE.

- *Alarm Color*: Enter a DWORD variable determining the alarm color of the element. This overwrites the value currently set by *Colors > color*. *Color* is used when the variable defined in *Toggle Color* is TRUE.

Look Variables: this element does not have the setting option *Fill attributes*.

Properties Image

Property	Value Description
Static ID	Identification of the image file for a static definition. Enter the ID of the image as used in the respective image pool (STRING); the name of the pool should be prefixed in order to make this entry unique (not necessary if the image is managed in the GlobalImagePool, because this one will be searched first anyway). For example: imagepool2.button_image. With click on <input type="text" value="..."/> , the Input Assistant opens and all available image pools with its images are listed.
Show frame	If activated, the referenced image will be displayed with a border.
Clipping	If this option together with Scale type fixed is activated, only that part of the image fitting into the frame will be displayed.
Transparent	If activated, the color defined in property Transparent color will be displayed transparently.
Transparent color	Button <input type="text" value="..."/> opens the color selection dialog for selecting a color which should be displayed transparently if option Transparent is activated.
Scale type	Specify here how the image should react on changes of the size of its frame: Isotropic: The image retains its proportions; thus even if height and width of the image frame are modified independently, the height-width-ratio of the image will be kept. Anisotropic: The image frame follows the size, so height and width of the image can be modified independently. Fixed: The original size of the image will be maintained regardless of the size of the image frame. Notice if option "Clipping" is also activated.
Horizontal alignment	Serves for explicitly maintaining the horizontal alignment with another element (as defined in the basic visualization) when the current visualization is used within a scaled frame element.. - Left - Centered - Right
Vertical alignment	Serves to explicitly keep the vertical alignment as defined in the basic visualization, in case the current visualization is used within a scaled frame element. - Top - Centered - Bottom

Table 8-65. Properties Image

Notas:

Scale type Isotropic: Regard the following if you want to maintain the elements alignment also within a scaled *Frame* element. For example, you have positioned a lamp centered above a switch and it should remain in this centered horizontal alignment, even if the size of the frame has changed. In order to avoid undesired horizontal or vertical shifts, explicitly define the Horizontal resp. Vertical alignment *Centered*.

Horizontal alignment: Only available if option *Expert* is activated for the Properties editor and if the image is of Scale type *Isotropic*.


Vertical alignment: Only available if option *Expert* is activated for the Properties editor and if the image is of Scale type *Isotropic*.

Bitmap ID Variable

Property	Value Description
Bitmap ID	Defines a project variable for the image ID.

Table 8-66. Properties Bitmap ID Variable

Input Assistant

The *Input Assistant* for images from image pools opens automatically if a visualization element *Image* is inserted in a visualization for the first time. You can get this dialog as well if you click in the image property view in the value field of Static ID and then perform .

Text Search

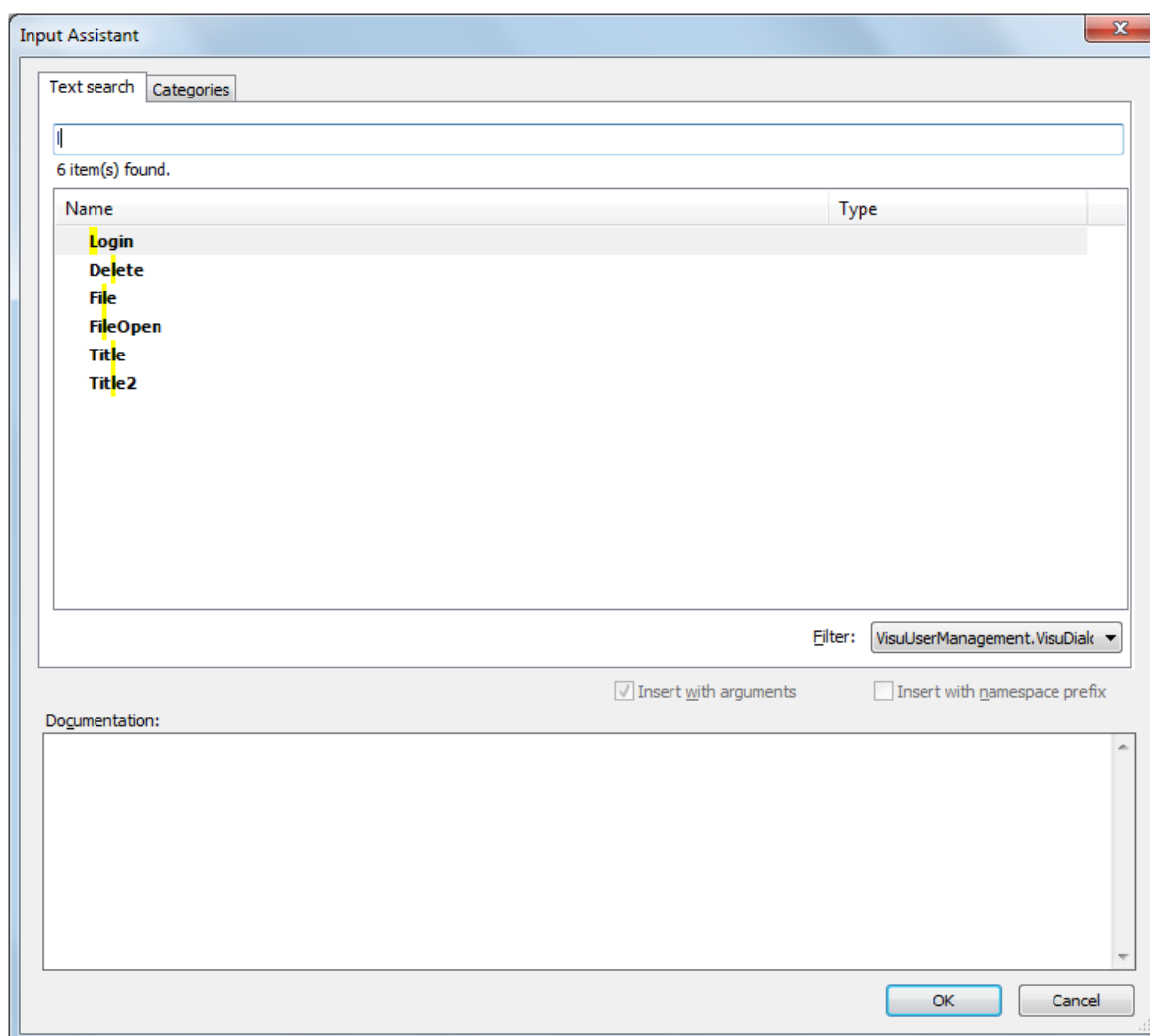


Figure 8-164. Input Assistant when assigning an image

First of all, set the *Filter* by selecting the desired image pool in the selection list. Then enter the name or a part of the name of the searched image file in the inline editor above. Immediately, all image files, which match with the text in the inline editor, are listed. Select one image from that and finish the dialog with [OK]. Or you can perform a double-click on the desired image. After you have

closed the dialog, this image is fully referenced in the property Static ID with <name of image pool>.<name of image>.

Categories

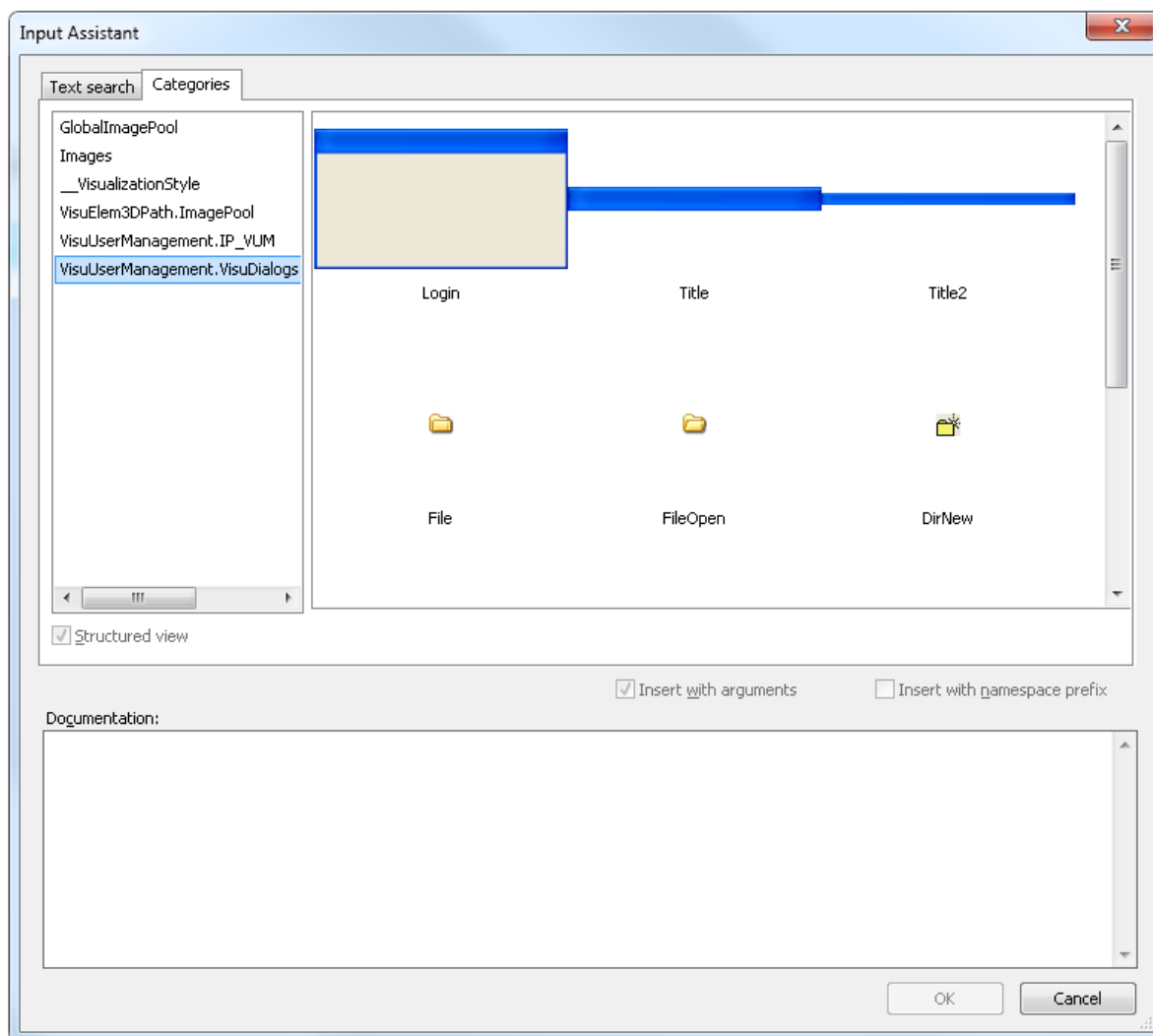


Figure 8-165. Input Assistant when assigning an image

Categories: all image pools of a project are listed in the left column. Select one of them. Then the images of the pool are listed in preview. Select one image from that and finish the dialog with [OK]. Or you can perform a double-click on the desired image. After you have closed the dialog, the image is fully referenced in the property Static ID with <name of image pool>.<name of image>.

Frame



Figure 8-166. Frame

Reference to Another Visualization.

Basically a visualization can be inserted in another visualization, thus getting *referenced*. For this purpose use the *frame* element which can enclose one or several references to visualizations. Which visualizations are currently available for getting inserted depends on their positions within the views POUs or Devices in relation to the current visualization. See **Frame Selection**.

Provide an Interface for Placeholders

A visualization can provide an interface to be opened with the **Basics of Interface Editor** command, where you can define input variables usual for a function block. These inputs work as placeholders. In an instance (reference) of the visualization, they must be replaced by values or expressions for the specific usage in the local object. The replacement is to be done in the *Properties* view of the frame element containing the visualization instance: *References > References*. Mind that the inputs of a visualization instance assigned to a certain application must be assigned to valid application variables.

Toggle Between Different Visualizations

A frame (like frames known from for example Internet pages) defines a partial area of the current visualization which can contain several other visualizations, between one can switch in online mode. Configure the inputs on these visualizations in a way that they each effect the display of a particular visualization in the frame. The dialog **Input Configuration** for this purpose provides the option *Switch Framevisualization*. Regard the possibility to define mouse actions with help of the property **Input Configuration**.

To define which visualizations should be displayed within a frame, open the *Frame Configuration* dialog by selecting the frame and perform command **Frame Selection**. Here all available visualizations will be offered and you can select the desired ones to get them inserted in the frame. A *filled* frame element then also in offline mode will show the contained visualization(s)..

Toggle Between Visualizations

A visualization contains three buttons "1", "2", "3" and a frame having got assigned the visualizations "Visu1", "Visu2" and "Visu3". The buttons are configured in a way (input: OnMouse-actions: change frame visualization) that activating a button will display the assigned visualization. Thus in one and the same visualization element various other visualizations and a *switch board* can be displayed.

Properties of Element Frame

The element *Frame* have all configuration settings presented in section Properties, but with some special features, which are listed in sequence:

Colors: this element has options a little different from those in the Table 8-49:

- *Color*: Select a base color for the element filled with when the element is in normal state. In case of the variabe defined in *Color variables* > *Toggle color* is FALSE, the element is in normal state.
- *Alarm Color*: Select a color, the element is filled with when the element goes in alarm state in case of the variabe defined in *Color variables* > *Toggle color* is TRUE.

Element Look: this element does not have the configuration option *Fill attributes*.

Color Variables: this element has options a little different:

- *Color*: Enter a DWORD variable determining the color of the element. This overwrites the value currently set by *Colors* > *color*. *Color* is used when the variable defined in *Toggle Color* is FALSE.
- *Alarm Color*: Enter a DWORD variable determining the alarm color of the element. This overwrites the value currently set by *Colors* > *color*. *Color* is used when the variable defined in *Toggle Color* is TRUE.

Look Variables: this element does not have the setting option *Fill attributes*.

Frame Properties

Property	Value Description
Clipping	If this option together with "Scale type" = fixed is activated, then only that part of the visualization fitting into the frame will be displayed. The option is grayed if "Scale type" = Fixed and scrollable.
Show frame	If activated, the referenced visualization will be displayed with a border.
Scale type	Specify here how the frame should react on changes of the size: <ul style="list-style-type: none"> - Isotropic: the referenced visualization retains its proportions; thus even if height and width of the frame are modified independently, the heigh-width-ratio of the visualization will be kept. - Anisotropic: the frame follows the size, so height and width of the referenced visualization can be modified independently. - Fixed: the original size of the visualization will be maintained regardless of the size of the frame. Notice if option "Clipping" is also activated. - Fixed and scrollable: use this option then the referenced visualization is displayed without scaling. If it is larger than the window area of the frame, the frame is fitted with scrollbars used for moving the displayed area of the visualization.
Deactivate the background drawing	To optimize the performance of the visualization the non-animated elements of a frame element are drawn as a background bitmap. This may lead to a display of the elements in an unexpected order. To avoid this behavior the function can be deactivated.
References	The command "Configure..." opens the dialog for the "Frame Selection". Here the references to visualizations assigned to the associated element are listed

Table 8-67. Frame Properties

Nota:

Scale type Fixed and scrollable: If you want to set the position of the scrollbars by variable, use the properties *Scroll position variable horizontal* or *Scroll position variable vertical*.

Scrollbar Setting

It is highly recommended to use client specific variables. If one of the scroll position variables are changed or the scrollbar is moved by mouse, then only the frame of the affected client is updated. If not, all clients will be updated.

Property	Value Description
Scroll position variable, horizontal	Enter a variable containing the position of the horizontal scroll bar.
Scroll position variable, vertical	Enter a variable containing the position of the vertical scroll bar.

Table 8-68. Properties Scrollbar

Client Specific Variable for Scroll Position

Código IEC

```
PROGRAM MainPrg
VAR
    iScrollHor : ARRAY[0..20] OF INT;
    iScrollVer : ARRAY[0..20] OF INT;
END_VAR
```

Properties set for the Frame Element

Property	Variable
Scroll position variable, horizontal	MainPrg.iScrollHor[CURRENTCLIENTID]
Scroll position variable, vertical	MainPrg.iScrollVer[CURRENTCLIENTID]

Table 8-69. Scrollbar Variables

Switch Frame Variable

This property allows to switch visualizations of a frame.

Property	Value Description
Variable	Enter a integer variable, which value contains the ID of the displaying visualization. The ID of a visualization is defined by the order of the visualization selection list of the frame configuration in dialog "Frame configuration". For the first item in the list "Selected visualizations" the ID is 0, for the second the ID is 1 and so on.

Table 8-70. Properties References

Common Controls

The set of *Common Controls* elements is composed of types Integer, Combo Box Array, Tab Control, Button, Group Box, Table, Text Field, Scrollbar, Slider, Spin Control, Invisible Input, Progress Bar, Radio Button and Checkbox.

The configurations and informations about **User Inputs when Editing an Element e User Interface when Editing the Properties** remain the same presented to the objects of the Basic group.

Label

O element *Label* serve para adicionar rótulos, títulos e qualquer outro tipo de texto para uma Visualization.

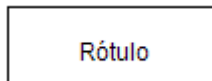


Figure 8-167. Label

Properties of Label

The element *Label* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**

But they have some special features, which are listed in sequence:

Texts: this element does not have the configuration option *Tooltip*.

State Variables: this element does not have the configuration option *Deactivate Inputs*.

Combo Box Integer

The element *Combo box integer* allows the user to choose one value from a drop down list. The index of the selected element will be written to a variable. A text list can be defined to supply the entries of the combobox or images from an image pool can be added.

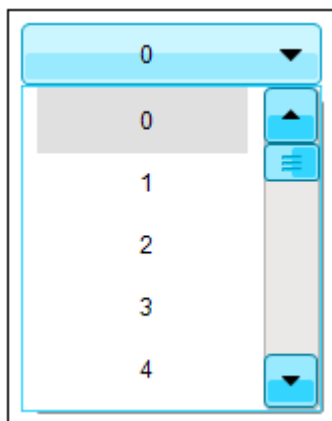


Figure 8-168. Combo Box Integer

Properties of Combo Box Integer

The element *Combo Box Integer* has the follow settings that are presented in section

Properties:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**

But they have some special features, which are listed in sequence:

Texts: The element *ComboBoxInteger* just has the option Tooltip.

ComboBoxInteger

Property	Value Description
Variable	Assign a enumerated variable containing the index of the selected list element..
Text List	Assign the name of the text list, type STRING, which contains and indicates all entries of the drop-down list
Image pool	Assign the name of the image pool, type STRING, which contains and indicates all images of the drop-down list.
Maximum Value	Define the maximum number of entries in the combo box.

Table 8-71. Properties ComboBoxInteger

Subrange

Property	Value Description
Use subrange	If this option is activated, only a subrange of the assigned list entries will be used.
Start Index	Define an integer value indicates the first entry of the entry list.
End Index	Define an integer value indicates the last entry of the entry list.

Table 8-72. Properties Subrange

Combo Box Table

The element *ComboBoxTable* allows the user to choose one value from a drop down list. The index of the selected row will be written to a variable.

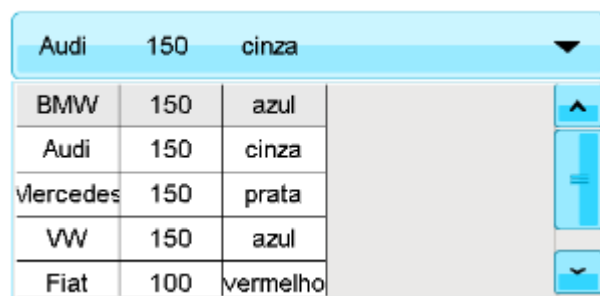


Figure 8-169. Combo Box Table

Properties of Combo Box Table

The element *ComboBoxTable* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**

But they have some special features, which are listed in sequence:

Texts: The option *Tooltip* is nonexistent for this element.

State Variables: The option *Deactivate Inputs* is nonexistent for this element.

ComboBoxTable

Property	Value Description
Variable	Assign a variable containing the index of the selected list row.
Data Array	Here the variable to be visualized is specified with its complete path. The structure of the variable determines the number of columns and rows of the table.

Table 8-73. Properties ComboBoxInteger

Columns


Property	Value Description
Column	Due to the structure of the variable defined in "Data array" the number of columns is identified and indexed in <n>. StringTabelle : ARRAY [0..2, 0..4] OF STRING := ['BMW','Audi','Mercedes','VW','Fiat','150','150','150','150','100','blue','grey','silver','blue','red'];: three columns are identified [0], [1] and [2].
Max. array index	An optional numeric variable can be configured for the visual table element to determine a dynamic maximum array index for use.
Row height	Height of table rows in pixels.
Scrollbar size	Width of vertical scrollbar in pixels.
Width	Width of the column in pixels.
Image Column	Checkbox to define the column as an image column. If it is activated images from the global image pool or any user-defined image pool will be displayed in this column of the table. Thereby the values of the cells of a table determine the ID of the image in the image pool.
Image configuration - Fill mode - Transparent - Transparent color	Fill mode: - Fill cell: The image follows the size of the cell without considering the height-width-ratio of the image. - Centered: The image will be centered in the cell and retains its proportions; thus even if height and width of the frame are modified independently, the height-width-ratio of the image will be kept. Transparent: If activated, the color defined in property "Transparent color" will be displayed transparently. Transparent color: enter a color or select one with help of button  que abrirá o dialog de seleção de cor. A cor definida será exibida de forma transparente se a opção Transparent estiver habilitada.
Text alignment of column	Alignment of columns: - Left - Centered - Right

Table 8-74. Properties Columns

Tab Control

The element *Tab Control* allows multiple forms to be contained within a single window, using tabs as a navigational widget for switching between sets of forms.

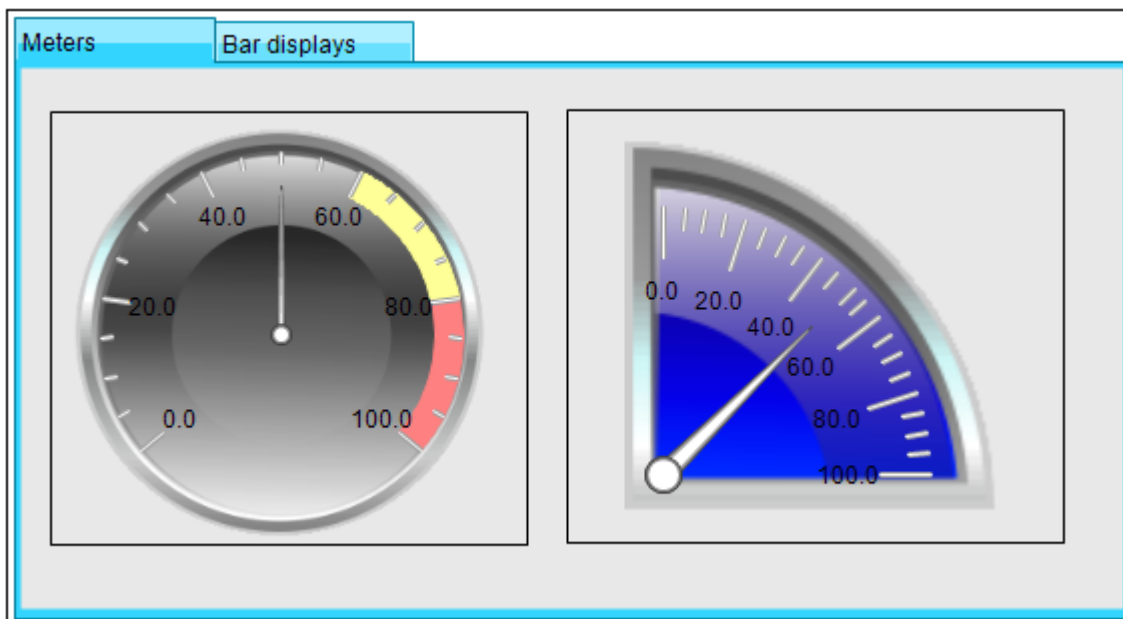


Figure 8-170. Tab Control

If the width of the tab control does not allow to display the tabs of all windows two navigation arrows will be displayed at the top right corner.

Properties of Tab Control

The element *Tab Control* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **State Variables**
- **Scrollbar Setting**
- **Switch Frame Variable**
- **Access Rights**

Tab Control

Property	Value Description
Tab width	Define the width of the tab in pixel.
Scale type	Specify here how the window should react on changes of the size: <ul style="list-style-type: none"> - Isotropic: the referenced Tab Control retains its proportions; thus even if height and width of the Tab Control are modified independently, the height-width-ratio of the visualization will be kept. - Anisotropic: the TabControl follows the size, so height and width of the referenced visualization can be modified independently. Fixed: the original size of the visualization will be maintained regardless of the size of the TabControl. - Fixed and scrollable: use this option then the referenced visualization is displayed without scaling. If it is larger than the window area of the frame, the frame is fitted with scrollbars used for moving the displayed area of the visualization. If you want to set the position of the scrollbars by variable, use the properties "Scroll position variable horizontal" or "Scroll position variable

	vertical".
Deactivate the background drawing	To optimize the performance of the visualization the non-animated elements of a frame element are drawn as a background bitmap. This may lead to a display of the elements in an unexpected order. To avoid this behavior the function can be deactivated.
References	The command "Configure..." opens the dialog for the "Frame Selection ". Here the references to visualizations assigned to the associated element are listed.

Table 8-75. Properties de Tab Control

Button

The element *Button* allows the user to trigger a function associated with the control. Optical adjustment is possible by assigning an image or by configuring a relief view.

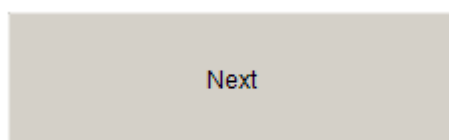


Figure 8-171. Button

Properties of Button

The element *Button* have all configuration settings presented in section Properties, but with some special features, which are listed in sequence:

Colors: this element has options a little different from those in the Table 8-49:

- *Color:* Select a base color for the element filled with when the element is in normal state. In case of the variabe defined in *Color variables > Toggle color* is FALSE, the element is in normal state.
- *Alarm Color:* Select a color, the element is filled with when the element goes in alarm state in case of the variabe defined in *Color variables > Toggle color* is TRUE.

Element Look: *Button* does not have this configuration option.

Color Variables: this element has options a little different:

- *Color:* Enter a DWORD variable determining the color of the element. This overwrites the value currently set by *Colors > color*. *Color* is used when the variable defined in *Toggle Color* is FALSE.
- *Alarm Color:* Enter a DWORD variable determining the alarm color of the element. This overwrites the value currently set by *Colors > color*. *Color* is used when the variable defined in *Toggle Color* is TRUE.

Look Variables: *Button* does not have this configuration option.

Properties Button


Property	Value Description
Button Height	Declaration of the height in pixel determines the relief-view of the button element:  .
Digital Variable	Enter a Boolean variable which will control the display of the button as "pressed" (variable = TRUE) resp. "not pressed" (variable = FALSE).

Table 8-76. Properties Button

Bitmap Info



Property	Value Description
Static ID	Here, the identifier of the image file, with which the button is to be visualized, is to specify static. The ID (STRING) is defined in the image pool. The name of the pool should be prefixed in order to make this entry unique except the image is managed in the GlobalImagePool, because this one will be searched first anyway. With click on  the Input Assistant opens and listing all available image pools with its images. For example: imagepool2.button_image
Scale type	Specify how the image should react on changes of the size of the button: - Isotropic: the image retains its proportions; thus even if height and width of the frame are modified independently, the height-width-ratio of the image will be kept. - Anisotropic: the image follows the size of the button, so height and width of the image can be modified independently. - Fixed: the original size of the image will be maintained regardless of the size of the button.
Transparent	If the option is activated, the bitmap will be displayed transparently with the color defined in "Transparent color".
Transparent color	Enter a transparency color applied to the bitmap if the option "Transparent" is activate. With click on  a color selection dialog opens.
Horizontal alignment	Define the horizontal alignment of the image. - Left - Centered - Right
Vertical alignment	Define the vertical alignment of the image. - Top - Centered - Bottom
Bitmap ID	Project variable defining the image ID.

Table 8-77. Properties Bitmap Info

Group Box

Elements can be dragged and dropped into a group box to form a optical unit. They can be nested several times.

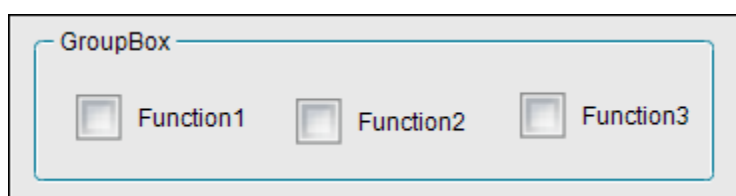


Figure 8-172. GroupBox

Properties of Group Box

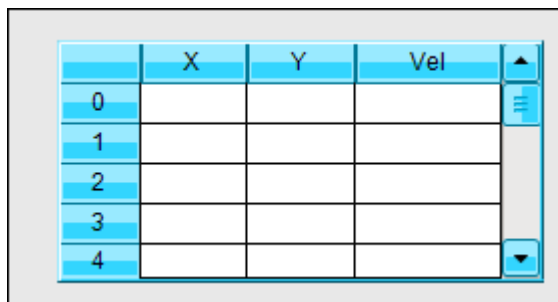
The element *Group Box* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**

But the property **Texts** does not have the configurations *Horizontal alignment* and *Vertical alignment*.

Table

A *Table* element can be inserted in a visualization in order to visualize one- or two-dimensional arrays, structures or the local variables of an POU.



	X	Y	Vel
0			
1			
2			
3			
4			

Figure 8-173. Selected Table with no Associated Data Array

Element specific user inputs:

- Change the column width directly by dragging the divider between two columns into the desired position while the cursor shape is + .
- A table column can be duplicated by pressing [Ctrl] and - while keeping pressed this key - a mouse click on the header of the respective column.
- A table column can be deleted by pressing [Ctrl] + [Shift] and - while keeping pressed these keys - a mouse click on the header of the column.

ATTENTION:

If you have changed the size of the variable on which the table is based on (array borders, number of structure components etc.) in the IEC code, but not yet updated the related table element in the visualization, you will get an appropriate compile message.

One way is to open the properties configuration and update the table for example by a [Return] in the *Data Array Value Field*.

A much more convenient way however is to perform a doubleclick on the error message. This will open a message box, where you can say Yes or No to an immediate update of the table configuration.

Properties of Table

The element *Table* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Texts**
- **Dynamic Texts**
- **Font Variables**
- **State Variables**
- **Access Rights**

But they have some differences, which are listed in sequence:

Texts: The element *Table* does not have the configuration options *Text e Tooltip*.

Table

Property	Value Description
Data array	Here the variable to be visualized is specified with its complete path. The structure of the variable determines the number of columns and rows of the table. If the number or array elements of the variable varies, use "Max. array index". For an update of the table in case the array or structure variable has changed size, use [Return] in the Data array Value field.
Max. array index	Assign a numeric variable to determine a dynamic maximum array index.


Table 8-78. Properties Table

ATTENTION:

If the *Data array* value gets changed by assigning a different variable, the other table properties will be reset to default.

Columns

The *Table* element visualizes a variable in table view. The index of the array elements resp. the components of a structure are displayed in one column resp. one row. Two-dimensional arrays or arrays of structures are displayed in multiple columns. Here, you can define the look of the table columns in which the values of the particular array elements / structure component / variable will be displayed. For each of the columns associated to index [*<n>*] a special configuration is provided.

Property	Value Description
Column - [n]	Due to the structure of the variable defined in "Data array" the number of columns is identified and indexed in <n>. var_data_array: ARRAY [2..8] OF INT: a single column is identified Column [0] var_data_array: ARRAY[1..3][1..10] OF INT: three columns are identified [0], [1] and [2]
Show row header	Checkbox to switch on or off the display of the labeling of rows in form of associated row indices of the array displayed.
Show column header	Checkbox to switch on or off the display of the labeling of columns defined by "Column header".
Row height	Height of table rows in pixels.
Row header width	Width of column containing the row headers in pixels.
Scrollbar size	Width of vertical and height of horizontal scrollbar in pixels.
Column header	By default the header displays the name of the array together with the index associated to the column. The header is modified by entering a new title here.
Width	Width of the column in pixels.
Image column	Checkbox to define the column as an image column. If it is activated images from the global image pool or any user-defined image pool will be displayed. Thereby the values of the cells of a table determine the ID of the image defined the image pool.
Image configuration - Fill mode - Transparent - Transparent color	Fill mode: - Fill cell: The image follows the size of the cell without considering the height-width-ratio of the image. - Centered: The image will be centered in the cell and retains its proportions; thus even if height and width of the frame are modified independently, the height-width-ratio of the image will be kept. If "Transparent" is activated, the color defined in property "Transparent color" will be displayed transparently. "Transparent color": enter a color or select one with help of button  opens the color selection dialog for selecting a color. It is displayed transparently if option "Transparent" is activated.
Text alignment of headline	Alignment of column header: - Left - Centered - Right
Use template	With activating this checkbox, a visual element of type Rectangle, Rounded rectangle or Ellipse will be inserted in every field of the table column associated

	to the index. The table properties list is extended automatically with the element properties of the chosen element under "Template".
Text alignment of headline from template	Activating this checkbox will only take effect if also the checkbox "Use template" is activated. If activated, the settings made within the properties for font (size) and alignment will be applied for the associated column header as well.
Template	This entry only exists if "Use template" is activated. Under "Template" the properties of the element assigned to the table fields are listed and can be edited as described in Rectangle, Round Rectangle and Ellipse.

Table 8-79. Properties Columns

Selection

Property	Value Description
Selection color	Define a cor de preenchimento usado para as células selecionadas.
Selection type	Defines which part of the table will be selected by clicking on one table cell: <ul style="list-style-type: none"> - No selection - Cell selection: only that cell will be selected. - Row selection: selection of the row containing the cell. - Column selection: selection of the column containing the cell. - Row and column selection: selection of row and column containing the cell.
Frame around selected cells	Checkbox to switch on or off a frame around the selected part.
Variable for selected column	Variable of type INT containing the array index of the column of the selected cell. In case the data array is a structure/POU, the structure components will be enumerated sequentially starting with 0. Be aware that the value only represents the correct position within the array/structure/POU if no column is suppressed from being displayed within the table.
Variable for selected row	Variable of type INT containing the array index of the row of the selected cell.
Variable for valid column selection	Boolean variable being TRUE if "Variable for selected column" contains a valid value.
Variable for valid row selection	Boolean variable being TRUE if "Variable for selected row" contains a valid value.

Table 8-80. Properties Selection

Assigning Variable to Table Element

First of all, the variable to be visualized must be specified. For this purpose in the element properties enter its complete path in property *Data array*.

The structure of this variable determines the number of columns and rows of the table. The rows of the table represent the index of the first dimension of the specified array. The columns represent the second index of the array in case of two-dimensional arrays or they represent the particular components of a structure or the local variables of a POU. So, for example, a one-dimensional array with elements of basic type will be represented with help of only one column, a structure will be represented with help of one single row, whereas an array with elements of structure type will be represented in a table having as many rows as array elements and as many columns as structure components.

Then configuration of the table with help of its further properties is possible.

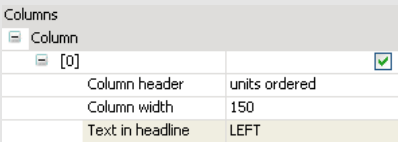
Assigned Data and the Resulting Table

Declaration of Variable	Resulting Table																																			
arrDim1: ARRAY[2..5] OF INT;	<table border="1"> <thead> <tr> <th></th> <th>arrDim1[INDEX]</th> </tr> </thead> <tbody> <tr> <td>2</td> <td></td> </tr> <tr> <td>3</td> <td></td> </tr> <tr> <td>4</td> <td></td> </tr> <tr> <td>5</td> <td></td> </tr> </tbody> </table> <p>Note that the indices in the row header are equal to the indices determining the first dimension of the array; so the numeration starts with "2" in this example.</p>		arrDim1[INDEX]	2		3		4		5																										
	arrDim1[INDEX]																																			
2																																				
3																																				
4																																				
5																																				
arrDim2 : ARRAY[0..2,0..3] OF INT;	<table border="1"> <thead> <tr> <th></th> <th>arrDim2[0,INDEX]</th> <th>arrDim2[1,INDEX]</th> <th>arrDim2[2,INDEX]</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td></td> <td></td> <td></td> </tr> <tr> <td>3</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		arrDim2[0,INDEX]	arrDim2[1,INDEX]	arrDim2[2,INDEX]	0				1				2				3																		
	arrDim2[0,INDEX]	arrDim2[1,INDEX]	arrDim2[2,INDEX]																																	
0																																				
1																																				
2																																				
3																																				
TYPE MyStruct : STRUCT iNo: INT; bBool: BOOL; sText:STRING; END_STRUCT END_TYPE varStruct: MyStruct;	<table border="1"> <thead> <tr> <th></th> <th>varStruct.iNo</th> <th>varStruct.bBool</th> <th>varStruct.sText</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		varStruct.iNo	varStruct.bBool	varStruct.sText	0																														
	varStruct.iNo	varStruct.bBool	varStruct.sText																																	
0																																				
arrStruct: ARRAY[0..3] OF MyStruct;	<table border="1"> <thead> <tr> <th></th> <th>arrStruct[INDEX].iNo</th> <th>arrStruct[INDEX].bBool</th> <th>arrStruct[INDEX].sText</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td></td> <td></td> <td></td> </tr> <tr> <td>3</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		arrStruct[INDEX].iNo	arrStruct[INDEX].bBool	arrStruct[INDEX].sText	0				1				2				3																		
	arrStruct[INDEX].iNo	arrStruct[INDEX].bBool	arrStruct[INDEX].sText																																	
0																																				
1																																				
2																																				
3																																				
varFB: TON;	<table border="1"> <thead> <tr> <th></th> <th>varFB.IN</th> <th>varFB.PT</th> <th>varFB.Q</th> <th>varFB.ET</th> <th>varFB.M</th> <th>varFB.StartTime</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		varFB.IN	varFB.PT	varFB.Q	varFB.ET	varFB.M	varFB.StartTime	0																											
	varFB.IN	varFB.PT	varFB.Q	varFB.ET	varFB.M	varFB.StartTime																														
0																																				
arrFB: ARRAY[0..3] OF TON;	<table border="1"> <thead> <tr> <th></th> <th>arrFB[INDEX].IN</th> <th>arrFB[INDEX].PT</th> <th>arrFB[INDEX].Q</th> <th>arrFB[INDEX].ET</th> <th>arrFB[INDEX].M</th> <th>arrFB[INDEX].StartTime</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>3</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		arrFB[INDEX].IN	arrFB[INDEX].PT	arrFB[INDEX].Q	arrFB[INDEX].ET	arrFB[INDEX].M	arrFB[INDEX].StartTime	0							1							2							3						
	arrFB[INDEX].IN	arrFB[INDEX].PT	arrFB[INDEX].Q	arrFB[INDEX].ET	arrFB[INDEX].M	arrFB[INDEX].StartTime																														
0																																				
1																																				
2																																				
3																																				

Table 8-81. Assigned Data and the Resulting Table

Configuration with Properties and the Resulting Table

Beside some basic properties like for example size, position or line width of the table you may adjust a multitude of table specific properties a selection of which shall be studied enlarging the example arrStruct . The effect of the different properties are shown step by step in the table below..

Changed Properties	Resulting Table																				
<p>First of all we change the standard entries in the column headers into more significant titles, adjust the width of the columns and determine the alignment of the titles in the header; for example for the first column.</p>  <p>There are two ways to change the column width: 1. In the element properties enter the desired value for Column > Width , or 2. Directly In the header line of the table element perform a mouse-click on the divider between two columns and draw it to the desired position</p>	<table border="1"> <thead> <tr> <th></th> <th>units ordered</th> <th>on stock?</th> <th>product code</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td></td> <td></td> <td></td> </tr> <tr> <td>3</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		units ordered	on stock?	product code	0				1				2				3			
	units ordered	on stock?	product code																		
0																					
1																					
2																					
3																					

We set the height of the rows to 20 pixels and switch off the row header, so that the first column displaying the indices of the array is missing now. The picture at right also shows the enlarged width ('Scrollbar size') of the scrollbar that appears as soon as the size of the table element is shrunk down under the sum of row heights resp. column widths.


Columns	
+ Column	
Row header	<input type="checkbox"/>
Column header	<input checked="" type="checkbox"/>
Row height	20
Row header width	15
Scrollbar size	40

units ordered	on stock?	product

We adjust the size of the table so that the scrollbars disappear again: the height equals the row height multiplied by the number of rows (+ 1 in case of column header switched on), the width equals the sum of the particular column widths and the row header width (in case of row header switched on). In general a small bonus (of for example 1) has to be added to these numbers.

Position	
X	21
Y	221
Width	486
Height	101

Text properties	
HorizontalAlignment	HCENTER
VerticalAlignment	VCENTER
Font	Tahoma; 12px

We change the style of lettering by selecting size, color and typeface in the dialog box opening on a click on  in the field entitled 'Font'. Note that the setting for the horizontal alignment ("Text alignment of headline") is only valid for the row header; the text in the table fields are aligned as prescribed by the column settings.

The online representation of the table has now been given a new look:

	units ordered	on stock?	product code
0	250	TRUE	DD32F7098367
1	480	FALSE	DD33F1782648
2	1500	TRUE	DD45G2637526
3	2500	TRUE	AD70F1829388

Note that the alignment of the text in the table fields follows the specific settings made for the columns.

The table fields can be edited column by column: For this purpose for a selected column activate the checkbox "Use template", whereon a folder entitled "Template" will expand within the column properties:

Use template	<input checked="" type="checkbox"/>
Use text alignment in headline	<input type="checkbox"/>
Template	
Type of element	VISU_ST_RECTANGLE
Colors	
Normalstate	
Framecolor	Coral
Fillcolor	Bisque
Alarmstate	
Framecolor	OrangeRed
Fillcolor	LightSalmon
Elementblock	
Linewidth	1
Fillattributes	B5_SOLID
Frameattributes	P5_DASH
Texts	
Text	%s
Tooltip	
Text properties	
HorizontalAlignment	RIGHT
VerticalAlignment	VCENTER
Font	Copperplate Gothic Light; 12px; style=Bold
Text variables	
Text Variable	PLC_PRG.arrStruct[INDEX].iNo

Three types of template elements are available: the rectangle is entered by default, what may be changed (into a rounded rectangle or a ellipse) by a click in this field. The related configuration possibilities reflect the element properties of the selected template element: as usual you can specify fill and frame color of the element in normal and alarm state that may also be toggled by an assigned variable. The alignment of the text within the column cells (except for the headers) will now follow the settings of the template.

In the example below a template has been used for the first and third column. The template of the first column corresponds to the settings demonstrated at left:

	units ordered	on stock?	product code
0	250	TRUE	DD32F7098367
1	480	FALSE	DD33F1782648
2	1500	TRUE	DD45G2637526
3	2500	TRUE	AD70F1829388

Note: Instead of the array element entered per default in the field 'Text variables', you may enter any other variable contained in the project whose value you want to be displayed. So, in particular, it is possible to change the ordering of the displayed array elements within the table.

Changing the columns can also be done in the table element itself. In the title line perform a mouse-click on the center of a column and while keeping the mouse-button pressed draw the column to the desired position

The behavior of the selected region is controlled by the section 'Selection':

Selection	
Selection color	HotPink
Apply to columns	Apply
Selection type	SEL_ROW
Frame around selected cells	<input checked="" type="checkbox"/>
Variable for selection X	PLC_PRG.SelectedPosX
Variable for selection Y	PLC_PRG.SelectedPosY
Variable for valid selection	PLC_PRG.IsSelected

The field "Selection type" prescribes what elements shall be selected by clicking on one table cell: as we have chosen the selection type Row selection all cells belonging to the same row as the one the click has been performed on will be classified as selected and included by a frame

If no columns are excluded from being displayed, the position of the selected cell within the table represents the indices belonging to the entry of the visualized array. These indices may then be written to the variables entered in the last three lines of the section "Selection". The third and fourth variable are of type Boolean and will be set to TRUE if the selection is valid concerning line resp. column.

The associated variables are set to the array indices corresponding to the selected cell:

Expression	Type	Value
arrDim1	ARRAY [2..5] OF INT	
arrDim2	ARRAY [0..2, 0..3] O...	
varStruct	MyStruct	
arrStruct	ARRAY [0..3] OF My...	
varFB	TON	
arrFB	ARRAY [0..3] OF TON	
iCounter	INT	745
iRow	INT	0
iCol	INT	7
SelectedPosX	INT	2
SelectedPosY	INT	1
IsSelected	BOOL	TRUE

Table 8-82. Configuration with Properties and the Resulting Table

Text Field

Within this element it is possible to display text that has been entered directly in the element properties or with help of a *Text variable*. In contrast to the rectangle, the frame can be shaded.

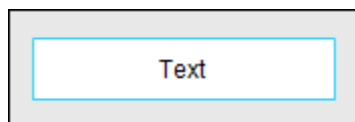


Figure 8-174. Textfield

Properties of Text Field

The element *Text Field* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Colors**
- **Element Look**
- **Texts**
- **Text Variables**

- **Color Variables**
- **State Variables**
- **Tap**
- **Toggle**
- **Hotkey**
- **Access Rights**

Shadow Type

Property	Value Description
Shadow Type	Type of the shading of the outline: - Deepened - No shadow - Raised - From style

Table 8-83. Properties Shadow Type

Scrollbar

This element adds a scrollbar to the visualization whose minimum and maximum value can be defined. The position of the actuator follows the value of the variable defined in Value.

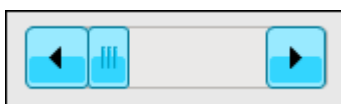


Figure 8-175. Scrollbar

Element Specific mouse Input

Por padrão, a barra de rolagem será exibida horizontalmente. Este element pode ser redimensionado com a ajuda do mouse. Assim que a altura for maior que a largura, a barra de rolagem é exibida verticalmente.

Properties of Scrollbar

The *Scrollbar* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Texts**
- **Text Variables**
- **Color Variables**
- **Dynamic Texts**
- **Font Variables**
- **State Variables**
- **Access Rights**

But they have some differences, which are listed in sequence:

Color Variables: this element has options a little different from those in the Table 8-49:

- *Color*: Select a base color for the element filled with when the element is in normal state. In case of the variabe defined in *Color variables* > *Toggle color* is FALSE, the element is in normal state.

- *Alarm Color*: Select a color, the element is filled with when the element goes in alarm state in case of the variable defined in *Color variables* > *Toggle color* is TRUE.

Scrollbar

Property	Value Description
Value	Assign a variable, whose value defines the current position of the actuator.
Minimum Value	Define the minimum value of the range.
Maximum Value	Define the maximum value of the range.
Page Size	Clicking in the scroll area, the position of the actuator is moved in the direction of the mouse. Here, define how much is moved (as a part of the range).
Orientation	The bar can be orientated in horizontal or vertical orientation. It results from the ratio of length to width and may not be edited here, but in the visualization editor by drag & drop
Running Direction	If the orientation is horizontal, choose between: <ul style="list-style-type: none"> - Left to right (da esquerda para direita) - Right to left (da direita para esquerda) If the orientation is vertical, choose between: <ul style="list-style-type: none"> - Bottom to top (de baixo para cima) - Top to bottom (de cima para baixo)

Table 8-84. Properties Scrollbar

Slider

The bar of the *Slider* can be moved by use of the mouse and the variable assigned to the slider will change its value within the defined scale values.

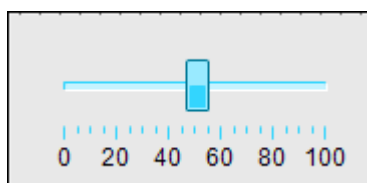


Figure 8-176. Slider

Element Specific Mouse Input

Per default the slider will be displayed horizontally. Having inserted this element, you may drag its form by help of the mouse. As soon as its height exceeds its width the scrollbar will appear vertically and the associated property *Bar* > *Orientation* is updated to *Vertical*.

Properties of Slider

The element *Slider* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **State Variables**
- **Access Rights**

Slider

Property	Value Description
Variable	Here a numeric variable has to be assigned containing the position of the handle of the slider.
Diagram Type	Here due to the orientation of the slider a diagram type can be chosen. If the orientation is vertical, choose between - Left: for a bar being left from the scale - Right: for a bar being right of the scale - Left and right: for a bar in the middle of two scales. If the orientation is horizontal, choose between - Top: for a scale being above the bar. - Bottom: for a scale being under bar. - Top and bottom: for a bar in the middle of two scales..
Orientation	The bar can be orientated in "horizontal" or "vertical" orientation. The orientation results from the ratio of length to width and may not be edited in the properties view but in the visualization window by drag and drop.
Running Direction	If the orientation is horizontal, choose between: - Left to right: "Scale start" is left. - Right to left: "Scale start" is right. If the orientation is vertical, choose between: - Bottom to top: "Scale start" is bottom - Top to bottom: "Scale start" is top.

Table 8-85. Properties Slider

Scale

Property	Value Description
Show scale	By default this checkbox is activated. Deactivating will fade out the scale.
Scale start	Value limiting the scaling range from below.
Scale end	Value limiting the scaling range from above.
Main scale	Distance between two scaling marks indicating coarse scale.
Sub scale	Distance between two scaling marks indicating fine scale. Can be set to 0 if no subscaling marks shall be displayed.
Scale format (C-syntax)	If a scale format like "%d" is configured, the scale shows the scale labels additionally

Table 8-86. Properties Scale

Spin Control

The element *Spin Control* allows to increase or decrease the value of a variable by a mouse click on the little arrow buttons on the right side of the element.

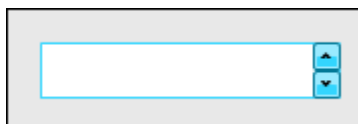


Figure 8-177. SpinControl

Properties of Spin Control

The element *SpinControl* has the follow settings that are presented in Properties:

- **General**
- **Position**

- **Color Variables**
- **State Variables**
- **Hotkey**
- **Access Rights**

Since the **C Color Variables** property only has the setting *Toggle Color*. In addition, the element *SpinControl* has the following property:

Spin Control

Property	Value Description
Variable	Enter a numeric variable which value will increase or decrease after an element user input
Number format	Enter a formatting sequence the variable value is displayed with. For example %l for a integer variable.
Interval	Definition of the interval (step width) by which the variable value is incremented or decremented according to the user input

Table 8-87. Properties Spin Control

Invisible Input

The element is displayed in the editor with a dashed line but in online mode it is invisible. The element behavior can be set via input configuration.

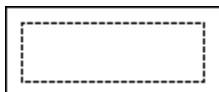


Figure 8-178. Invisible Input

Properties of Invisible Input

The element *Invisible Input* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **State Variables**
- **Tap**
- **Toggle**
- **Hotkey**
- **Access Rights**

Since the **State Variables** property only has the configuration option *Deactivate Inputs*.

Progress Bar

A progress bar conveys the progress of something. This requires a variable whose value is displayed as bar. The limits of the variable as well as the “Style” of the progress bar can be set.

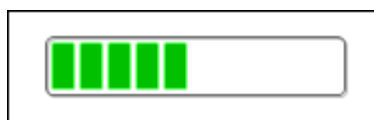


Figure 8-179. Progress bar

Properties of Progress Bar

The element *Progress Bar* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**

But they have some special features, which are listed in sequence:

Texts: The element *Progress Bar* has only the option *Text*.

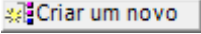
State Variables: O Elemento *Progress Bar* does not have the configuration *Deactivate Inputs*.

Progress Bar

Property	Value Description
Variable	Define the variable which value is to be displayed as progress bar.
Minimum value	Define here the minimum value of the variable.
Maximum value	Define here the maximum value of the variable.
Style	Choose one of following styles: - Blocks - Bar

Table 8-88. Properties Progress Bar

Radio Button

The radio button element allows to set any number of radio buttons by use of the  button and can be arranged in one or several columns.

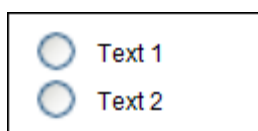


Figure 8-180. Radio button

Properties of Radio Button

The element *Radio Button* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **State Variables**
- **Access Rights**

Radio Button

Property	Value Description
Variable	Assign a variable the index of the selected radio button will be written to.
Number of columns	Defines the number of columns that will be displayed.
Radio button order	- Left to right: The radio buttons are listed line by line as long as the number of columns is reached - Top to bottom: The radio buttons are listed columnwise as long as the number of columns is reached
Radio button - Areas [<n>]	Create now: With click on this button, a further radio button is created in the editor and a further area is listed in the property view. A area for every radio button for gathering the settings is visible. [<n>]: The number indicates the area. With click on "Delete", the associated radio button with its settings ("Text", "Tooltip", "Line spacing in pixels") will be deleted.
Text	Here, the desired button name can be entered. Default value is "Radio button".
Tooltip	Here a text displayed as tool tip can be entered.
Line spacing in pixels	Here the space to the button above can be entered in pixels.

Table 8-89. Properties Radio Button

Checkbox

With a *Checkbox* a Boolean variable can be set or reset. If the tick is set, the variable is set to TRUE.



Figure 8-181. Check box

Properties of Checkbox

The element *Checkbox* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**

Texts: The element *Checkbox* only has the configuration options *Text* e *Tooltip*.

Measurement Controls

The set of *Measurement Controls* elements is composed of types Bar Display, Meter 90°, Meter 180°, Meter 360°, Potentiometer and Histogram.

The configurations and informations about **User Inputs when Editing an Element** e **User Interface when Editing the Properties** remain the same presented to the objects of the Basic group.

Properties Measurement Control

The Measurement Control set has a property group common for the most part of its elements:

Background

Property	Value Description
Background image	Select a color for the bar background: <ul style="list-style-type: none"> - Yellow - Red - Green - Blue - Gray

Table 8-90. Properties Background

Arrow

Property	Value Description
Arrow type	Select one of: <ul style="list-style-type: none"> - Normal arrow - Thin arrow - Wide arrow - Thin needle - Thin 3D arrow - Thin 3D needle
Color	Color the arrow shall be displayed with.
Angle range	Select a 90° sector: <ul style="list-style-type: none"> - Top right - Top left - Bottom left - Bottom right
Additional arrow	If the checkbox is activated, an additional arrow is visible on the scale in opposite of the needle.

Table 8-91. Properties Arrow

Scale


Property	Value Description
Sub scale position	The sub-scale can be displayed at the outer or inner radius of the scale ring: <ul style="list-style-type: none"> - Outside - Inside
Scale type	The scale can be displayed as follows: <ul style="list-style-type: none"> - Lines - Dots - Squares
Scale start	Value limiting the scaling range from below.
Scale end	Value limiting the scaling range from above.
Main scale	Distance between two scaling marks indicating coarse scale.
Sub scale	Distance between two scaling marks indicating fine, sub-scale. It can be set to 0 if no subscaling marks shall be displayed.
Scale line width	Define the width of the scale line in pixel.
Scale color	The color can be set via the selection list or via the standard dialog for color selection to be accessed by button 
Scale in 3D	Enable the option to get a three dimensional look of the scale.
Show scale	If no scale is desired, disable the option.
Frame inside	If this option is enabled, the scale ring is provided inside with a frame.
Frame outside	If this option is enabled, the scale ring is provided outside with frame.

Table 8-92. Properties Scale

Label



Property	Value Description
Label	Select: - Outside: the scale values are placed on outside position in relative to the scale. - Inside: the scale values are placed on inside position in relative to the scale
Unit	String that will be displayed below the arrow (M) or below the midpoint of the range(BD) (per default the string is thought to indicate the unit used for scaling)
Font	Font for unit and scaling range. Define the size for the font by select one of: - Default - Headline - Large - Title - Annotation With click on  a dialog for a user-defined setting of the font properties will open.
Scale format (C-syntax)	Use C-syntax for indicating the formatting of the scale labels. So entering the string "%3.2f s" in this edit field will cause the scale labels to be displayed with 3 digits, whereof two digits are located behind the decimal point and followed by the letter "s".
Max. text width of labels	Value limiting the maximum width of text labels. Usually the width of the labels will be adjusted automatically. Use this edit entry in case of failure of the automatic adjustment.
Text height of labels	Value defining the height of text labels. usually the height of the labels will be adjusted automatically. Use this edit entry in case of failure of the automatic adjustment.
Font color	The font color can be set via the selection list resp. via the standard dialog for color selection to be accessed by button  .

Table 8-93. Properties Label

Positioning

Property	Value Description
Needle movement	It defines an offset where the needle should be placed.
Label offset	Define a offset for the position of the label.
Unit offset	Define a vertical offset for the position of the text entered in Label > Unit .
Origin offset	Origin offset


Table 8-94. Properties Positioning

Color Areas

Property	Value Description
Color areas [<n>]	Durable color areas: Its effect is only visible in online mode. If the option is activated, the color areas are durably visible . If it is not activated, this color area the arrow is located in is visible. Create now: With click on this button, a further area is created and listed in the property view..
Begin of area	Value for limiting the area range from below.
End of area	Value for limiting the area range from above.
Color	Color the area shall be displayed with.

Table 8-95. Properties Color Areas

Nota:

Color areas: It is possible to remove an color area by pressing the button .

Bar Display, Bar Display Free Configurable

Here, a bar display can be added to the visualization. *Bar Display* has a pre-defined design in which the background color can be chosen. *Bar Display free configurable* allows to assign a image from the *GlobellImagePool* in properly as background image. That's the only difference.

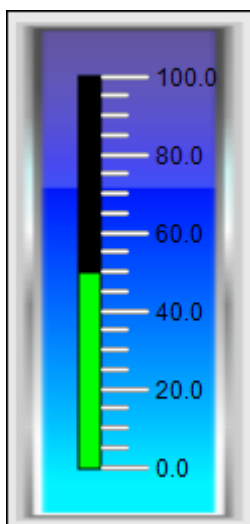


Figure 8-182. Bar Display

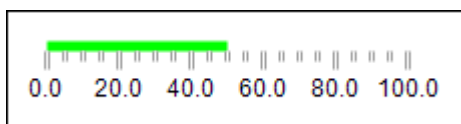


Figure 8-183. Bar Display free Configurable

Element Specific Mouse Input:

Por padrão, a barra será exibida horizontalmente. Após ter inserido este element poderá arrastar sua forma com a ajuda do mouse. Assim que sua altura é superior à sua largura, a barra aparecerá na vertical e sua propriedade *Bar > Orientation* é automaticamente ajustada para Vertical.

Properties of Bar Display

The element *Bar Display* has the follow settings that are presented in sections *Properties* and **Properties Measurement Control:**

- **General**
- **Position**
- **Access Rights**
- **Background**
- **Scale**
- **Label**
- **Color Areas**

But they have some differences, which are listed in sequence:

Background: If used the element *Bar display free configurable* this configuration option will be changed, and you can choose a background image from *GlobellImagePool* by typing the file name or ID.

Scale: the element does not have the configuration options *Sub scale position*, *Scale type*, *Show scale*, *Frame inside* and *Frame outside*.

Label: the element does not have the parameter *Label*

Color Areas: the element does not have the option *Durable color areas*.

Bar Display

Property	Value Description
Value	Assign a variable, whose value then is visualized as bar length.
Diagram type	The drop-down menu provides different possibilities for locating the bar with respect to the scaling: <ul style="list-style-type: none"> - Scale beside bar - Scale in bar - Bar in scale - No scale.
Orientation	The bar can be orientated in "horizontal" or "vertical" orientation. The orientation results from the ratio of length to width and may not be edited here but in the visualization window by grabbing any window edge of the element to stretch the window horizontally or vertically.
Running direction	If the orientation is horizontal, choose between: <ul style="list-style-type: none"> - Left to right - Right to left. If the orientation is vertical, choose between: <ul style="list-style-type: none"> - Bottom to top - Top to bottom.

Table 8-96. Properties Bar Display

Positioning

Property	Value Description
Horizontal offset	Here the distance in pixel from scale/bar to the horizontal element frame can be set.
Vertical offset	Here the distance in pixel from scale/bar to the vertical element frame can be set.

Table 8-97. Properties Positioning

Bar Color

Property	Value Description
Graph color	Define a color for the bar.
Bar background	By default the background color of the bar is white; activating the checkbox will change the background color to black.
Frame color	Color of frame surrounding bar display element in case of checkbox "Element frame" being activated.
Use color areas	If this checkbox is activated, the specified color areas will be displayed as they are defined under "Color areas".
Switch whole color	With activating the checkbox, the entire bar has to change its color if the variable value exceeds the start/end value of the scale.
Color range markers	Select between: <ul style="list-style-type: none"> - No markers - Marker forward - Marker backward

Table 8-98. Properties Bar Color

Meter 90°, 180° e 360°

With meter element for example a tachometer can be added to a visualization. The position of the needle is then adjusted to the value of the associated input variable. It has a preconfigured design where a color as background can be set.

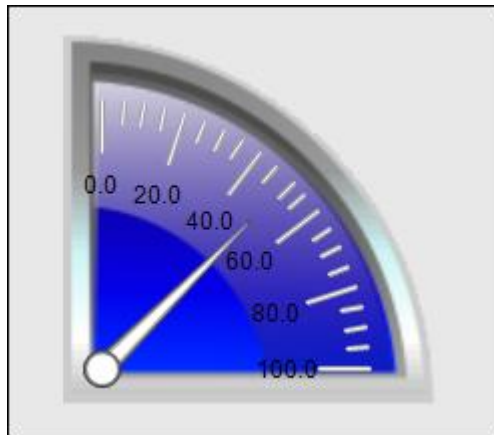


Figure 8-184. Meter 90°

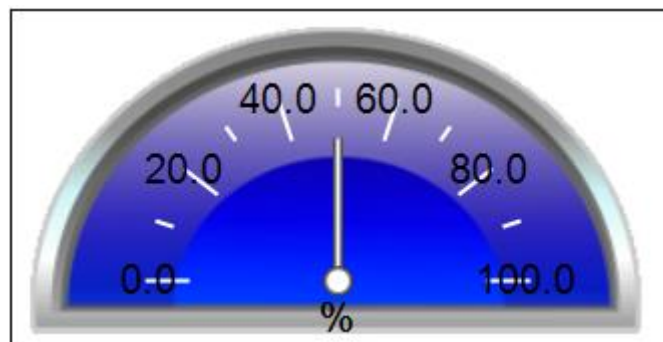


Figure 8-185. Meter 180°

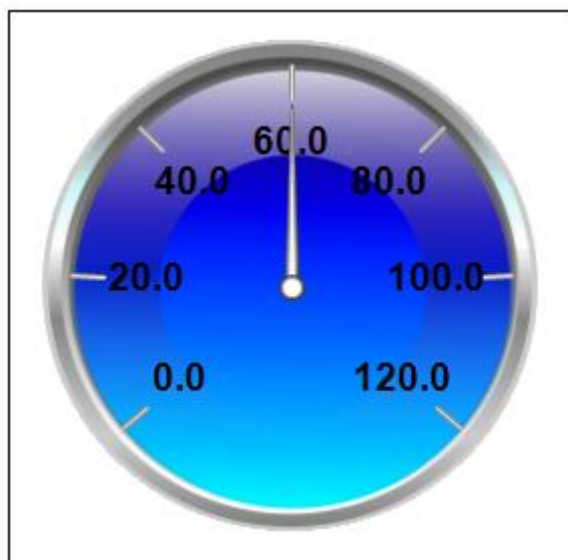


Figure 8-186. Meter 360°

Properties of Meter 90°, 180° and 360°

The element *Meter* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Access Rights**

Additionally these elements have the properties displayed in the **Properties Measurement Control** section, with the following features:

Positioning: The elements Meter 90° and 180° does not have the option *Origin offset*.

Meter 90°, 180° e 360°

Property	Value Description
Value	Assign a variable whose value then is visualized as deflection of the meter needle.

Table 8-99. Properties Meter

Potentiometer

Here a *Potentiometer* with pre-defined design can be added to the visualization. This control element displays a variable value by a arrow that can be moved via user input.



Figure 8-187. Potenciômetro

Properties of Potentiometer

The element *Potentiometer* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Access Rights**

Additionally these elements have the properties displayed in the **Properties Measurement Control** section, with the following features:

Background: the element *Potentiometer* have the configuration option *Transparency color*, where is possible to select a color to be transparent.

Arrow: the options *Angle range* and *Additional arrow* were changed by::

- *Arrow start:* The value listed here is interpreted as an angle in degree between zero point of the scale and the start of the scala in counter-clockwise direction. Zero point of the scale is "3 o'clock" . The angle ranges from 0° to 360°. Center of rotation is the X/Y position.
- *Arrow end:* The value listed here is interpreted as an angle in degree between zero point of the scale and the end of the scala in counter-clockwise direction. Zero point of the scale is "3 o'clock". Input of a negative value is allowed. Center of rotation is the X/Y position. "Arrow end" must be set in a way, that it is in clockwise direction right from "Arrow start". The angle can range from 0° to 350°.

Potentiometer

Property	Value Description
Variable	Here a numeric variable has to be assigned containing the position of the potentiometer.

Table 8-100. Properties Potentiometer

Histogram

This element adds a histogram view to the visualization, which displays a data array. There minimum and maximum values can be defined. For particular value ranges, background colors may be specified.

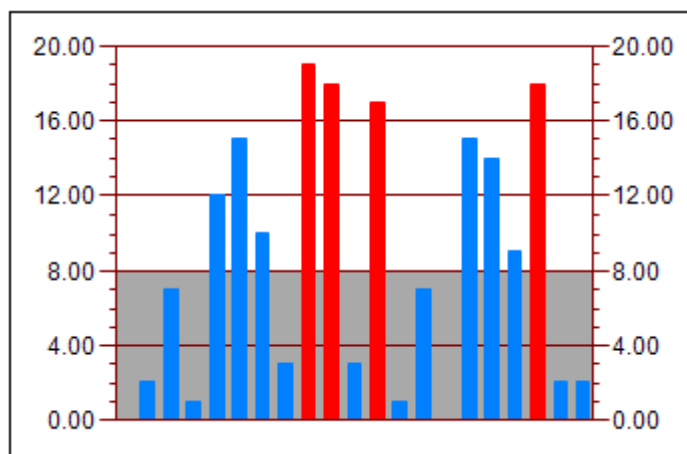


Figure 8-188. Histogram

Properties of Histogram

The element *Histogram* has the follow settings that are presented in sections **Properties** and **Properties Measurement Control**:

- **General**
- **Position**
- **Access Rights**
- **Scale**
- **Label**
- **Color Areas**

But they have some differences, which are listed in sequence:

Scale: this element does not have the configuration options *Sub scale position*, *Scale type*, *Show scale*, *Frame inside*, *Frame outside*, *Scale line width* and *Scale in 3D*.

Label: this element does not have the parameter *Label*

Color Areas: this element does not have the option *Durable color areas*.

Data Array

Property	Value Description
Data array	Assign an data array for visualization.
Use subrange	If this option is activated, only a subrange of the assigned data array will be used.
Start index	Indicates the first set of data of the assigned data array.
End index	Indicates the last set of data of the assigned data array.

Table 8-101. Properties Data Array

Histogram

Property	Value Description
Display type	- Bars - Lines - Curves
Show horizontal lines	If this option is activated, horizontal lines will be drawn on main scale.
Relative bar width	This value defines the relative width of the bars resp. lines in pixels

Table 8-102. Properties Histogram

Histogram Color

Property	Value Description
Graph color	Define a color for the graph
Alarm color: - Alarm value - Alarm condition - Alarm color	Alarm value: Define here a threshold for alarm. Alarm condition: If the current value of the array element fulfills the alarm condition, the alarm color is set. - Less: alarm is set, if the value is smaller than the alarm value. - More: alarm is set, if the value is greater than the alarm value. Alarm color: the single bar will be displayed with this alarm color.
Use color areas	If this checkbox is activated, the specified color areas will be displayed as they are defined under Color areas.

Table 8-103. Properties Histogram Color

Configuration of a Histogram Element

See in the following a sample of inserting and configuring a histogram element.

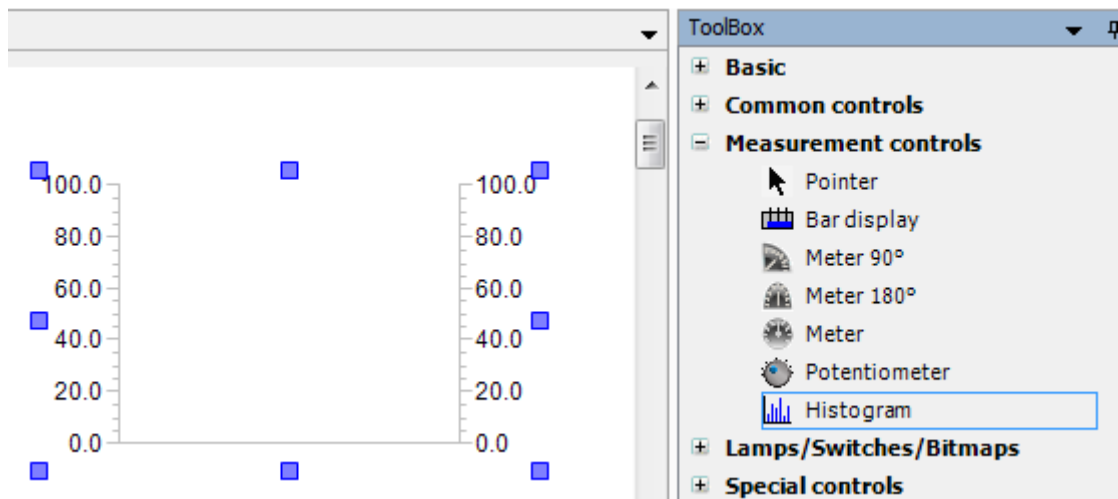


Figure 8-189. Example - Inserting a Histogram Element

The associated input data array - a array named Histogram in the example - has to be specified within the element properties *Data array* of the histogram element; after clicking in the edit field of property *Data array* you will find the button which can be used for selecting the data array by browsing your project. Take care to indicate the array by its entire path within the device tree.

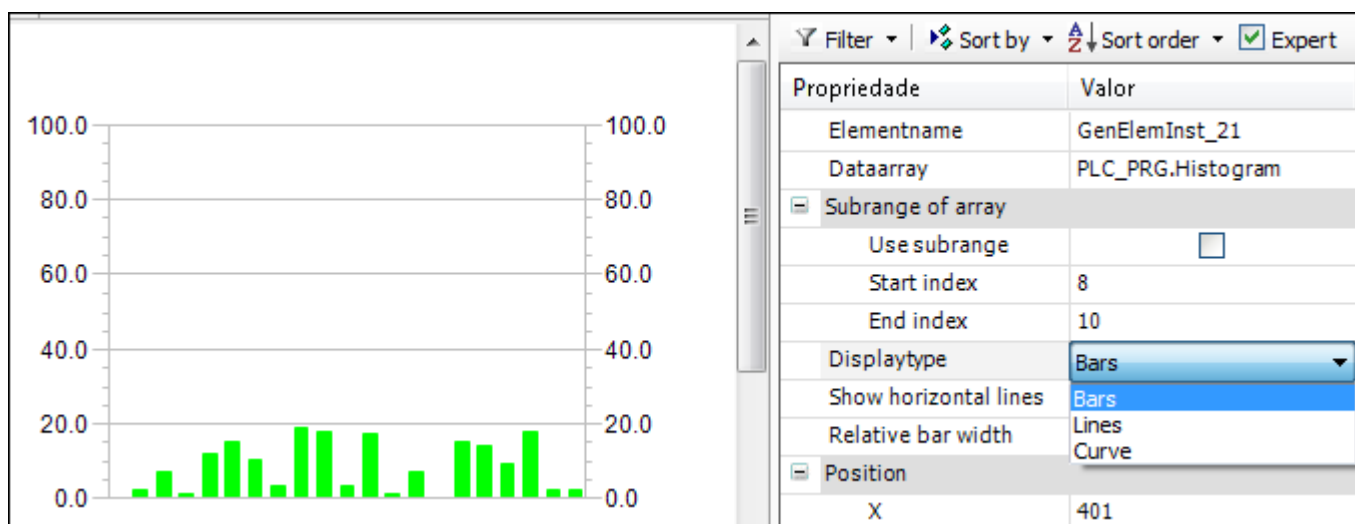


Figure 8-190. Example - Configuração do tipo de exibição.

Use subrange: This option is used to display not all elements of the array, but only the elements from index Start index to End index.

Display type: The data array can be displayed in three different styles:

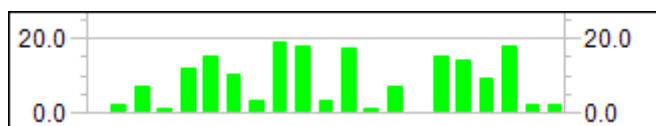


Figure 8-191. Bars

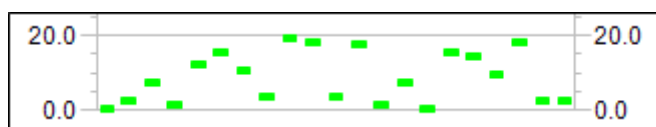


Figure 8-192. Lines

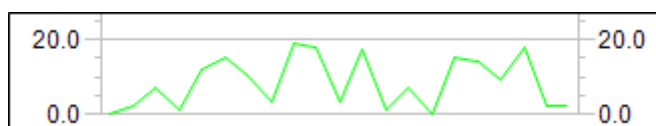


Figure 8-193. Curve

Show horizontal lines: If this option is activated, horizontal lines will be drawn on main scale.

Relative bar width: This value defines the relative width of the bars resp. lines.

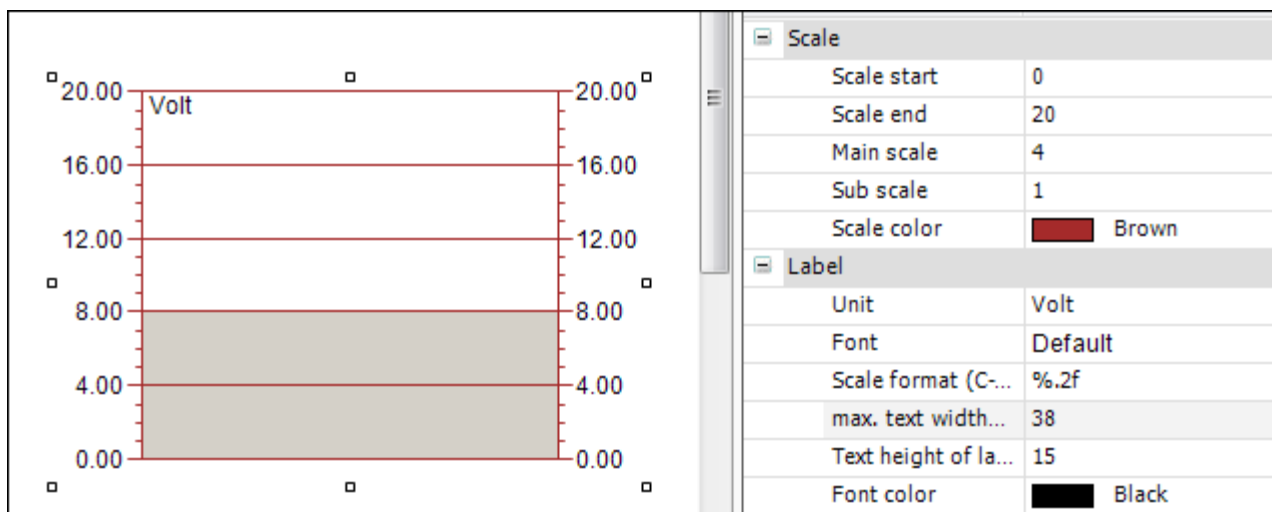


Figure 8-194. Example - Configuration of Histogram Scale.

The section **Scale** is provided for defining the numeric range of the scale and subdividing the scale by coarse and fine pitches. The range of the scale is limited by the *Scale start* from below and by the *Scale end* from above. Thus the start value has to be less than the end value. Both of the distances in *Main scale* and *Sub scale* may be set to 0 for disabling the display of the scaling: if the coarse scaling distance is set to 0 there will be no scaling displayed at all (regardless the setting made for the sub scale). If the sub scale is set to 0, only the coarse scale pitches will be drawn. To change the color of the scale perform a double click on the *Scale color* field to open a dialog for selecting colors.

In the section **Label** the layout of the scale values can be defined. The entry in the edit field *Unit* is thought for indicating the unit used within the scaling and displayed in to top left corner. Having chosen the appropriate (color of the) font, we adjust the format of the labels: The format for numeric scale values has to be indicated according to the syntax of the programming language C (use %d for integers and %.Xf for floating points, where X has to be replaced by the number of decimal places).

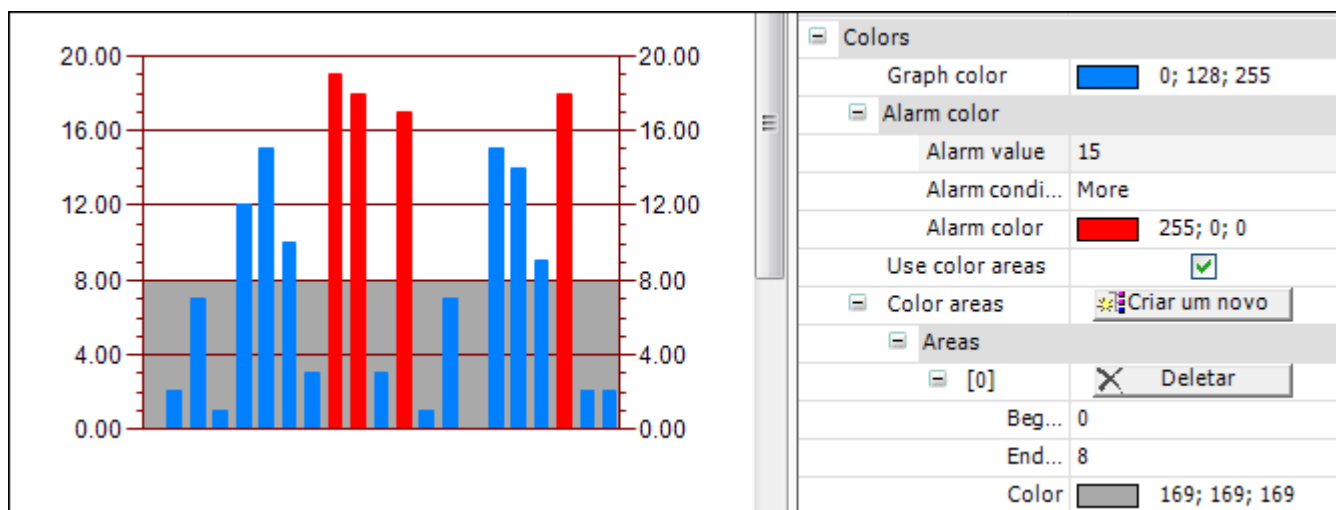




Figure 8-195. Example - Configuration of Histogram Colors

Finally we may adjust the coloring of the element within the sections **Histogram Color** and **Color Areas**:

First of all the histogram color may be selected. Within the subsection *Alarm color* an alarm color may be specified the single bar will be displayed in if the actual value of the array element fulfills the

alarm condition. This condition results from defining an alarm value, that may either not be exceeded (set condition to *More*) or not be underrun (set condition to *Less*).

Parts of the scale range can then be given a specific color by creating color areas with help of the button . The color areas are numbered consecutively in ascending order. Each color area will be provided with own edit field within the element properties:

Begin of the area and *End of the area* are provided to specify the subrange of the scaling range. The coloring can be selected from an drop-down menu and a color area once created can be deleted by clicking on the associated button .

Lamps, Switches, Bitmaps

The set of *Lamps, Switches, Bitmaps* elements is composed of types Image Switcher, Lamp, Dip Switch, Power Switch, Push Switch, Push Switch LED, Rocker Switch and Rotary Switch.

The configurations and informations about **User Inputs when Editing an Element e User Interface when Editing the Properties** remain the same presented to the objects of the Basic group.

Properties Lamps, Switches, Bitmaps

The Lamps, Switches, Bitmaps set has a property group common for the most part of its elements.

Background

Property	Value Description
Background image	Select a color: <ul style="list-style-type: none"> - Yellow - Red - Green - Blue - Gray

Table 8-104. Properties Background

Image settings

Property	Value Description
Isotropic type	Specify here how the element should react on changes of the size of the element "frame", when the current visualization is embedded into a frame. <ul style="list-style-type: none"> - Isotropic: the element retains its proportions, thus even if height and width of the frame are modified independently, the height-width-ratio of the image will be kept. - Anisotropic: the element follows the size of the frame, so height and width of the image can be modified independently. - Fixed: the original size of the element will be maintained regardless of the size of the element.
Horizontal alignment	Serves for explicitly maintaining the horizontal alignment with another element (as defined in the basic visualization) when the current visualization is used within a scaled frame: <ul style="list-style-type: none"> - Left - Centered - Right
Vertical alignment	Serves for explicitly maintaining the vertical alignment with another element (as defined in the basic visualization) when the current visualization is used within a scaled frame: <ul style="list-style-type: none"> - Top - Centered - Bottom

Table 8-105. Properties Image settings

Notas:

Expert: Setup option in the top right corner of the properties editor.

Horizontal e Vertical Alignment: Only available if option *Expert* is activated for the Properties editor and if the image is of type *Isotropic*.

Isotropic type Example: Regard the following if you want to maintain the elements' alignment also within a scaled frame. For example you have positioned a lamp centered above a switch and it should remain in this centered horizontal alignment, even if the frame size gets changed. In order to avoid undesired horizontal or vertical shifts, explicitly define the Horizontal resp. Vertical alignment *Centered*.

Image Switcher

The *Image Switcher* can be used to switch from Image-off to Image-on on user input. The state of the Boolean variable is set analogously. With a mouse click on the element the image as well as the assigned variable will be changed to Image-on and value TRUE. Due to the setting in *Element behavior*, Image-off will then displayed again when either the mouse button is released or when the next mouse click is set.

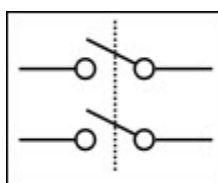


Figure 8-196. Example Image off

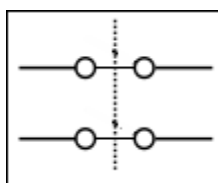


Figure 8-197. Example Image on

Properties of Image Switcher

The element *Image Switcher* has the follow settings that are presented in sections Properties and **Properties Lamps, Switches, Bitmaps:**

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**
- **Background**
- **Image settings**

But they have some differences, which are listed in sequence:

Texts: element *Image Switcher* only has the option *Tooltip*.

Image settings: element to *Image Switcher* has more properties:

- *Image on*: Enter the ID of a image file as used in the respective image pool (STRING). It is used as Image on.
- *Image off*: Enter the ID of a image file as used in the respective image pool (STRING). It is used as Image off.

Image Switcher

Property	Value Description
Variable	Boolean variable which state is changed after an user input. Depending of the value of "Element behavior" the variable is TRUE as long as the mouse key is pressed (Image tapper) or the value changes with every mouse click (Image toggler).
Element behavior	- Image toggler: The image and the variable value switch with every mouse click. - Image tapper: Image-on is displayed and variable is TRUE as long as the mouse key is pressed.
Tap FALSE	This option is only available if the "Element behavior" is set to "Image tapper". If not activated (default) a pressed mouse key will set the assigned variable to TRUE; otherwise the variable is set to FALSE.

Table 8-106. Properties Image Switcher

Lamp

The *Lamp* lights up when the variable assigned in *Variable* is set. With the property *Background image* the color of the lamp can be selected.

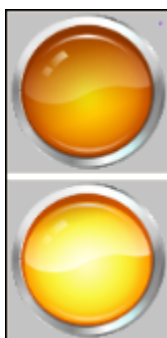


Figure 8-198. Lamp

Properties of Lamp

The element *Lamp* has the follow settings that are presented in sections *Properties* and **Properties Lamps, Switches, Bitmaps**:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**
- **Background**
- **Image settings**

But they have some differences, which are listed in sequence:

Texts: the elemento *Image Switcher* only has the configuration option *Tooltip*.

Lamp

Property	Value Description
Variable	Boolean variable which state is used to visualize the state of the lamp. With TRUE the lamp is visualized as turned on and looks bright.

Table 8-107. Properties Lamp

Dip Switch, Power Switch, Push Switch, Push Switch LED, Rocker Switch

The *Switch Elements* can be used to change the value of a Boolean variable.



Figure 8-199. Dip switch, Power switch, Push switch, Push switch LED, Rocker switch

Properties of Switch Elements

The *Switch Elements* has the follow settings that are presented in sections **Properties** and **Properties Lamps, Switches, Bitmaps**:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**
- **Background**
- **Image settings**

But they have some differences, which are listed in sequence:

Texts: the elemento *Image Switcher* only has the configuration option *Tooltip*.

Switch

Property	Value Description
Variable	Variável booleana cujo valor é alterado após uma entrada do usuário. Dependendo do valor do parâmetro "Element behavior", a variável receberá o valor TRUE enquanto o botão do mouse estiver pressionado (Image tapper) ou a cada clique do mouse (Image toggle).
Element behavior	- Image toggler: the switch is displayed as tuned-on and the variable value is TRUE as long as the mouse key is pressed. - Image tapper: the display of the switch and the variable value toggles with every mouse click.

Table 8-108. Properties Switch

Rotary Switch

The switch elements can be used to change the value of a boolean variable.



Figure 8-200. Rotary Switch

Properties of Rotary Switch

O element *Rotary Switch* has the follow settings that are presented in sections **Properties** and **Properties Lamps, Switches, Bitmaps**:

- **General**
- **Position**
- **Texts**
- **State Variables**
- **Access Rights**
- **Background**
- **Image settings**

But they have some differences, which are listed in sequence:

Texts: the elemento *Image Switcher* only has the configuration option *Tooltip*.

Rotary Switch

Property	Value Description
Variable	Boolean variable which value is changed after an user input. Depending of the value of "Element behavior" the variable is TRUE as long as the mousekey is pressed (Image tapper) or the value changes with every mouse click (Image toggle).
Element behavior	- Image toggler: The Rotary Switch is displayed as turned on and the variable value is TRUE as long as the mouse key is pressed. - Image tapper: Variable value switched with every mouse click.
Orientation	- At top: the switch is oriented upward. - At side: the switch is turned to the side.
Color change	If the checkbox is deactivated, the switch lights up when it is turned on. If the checkbox is activated, the switch doesn't light up. Even when it's turned on.

Table 8-109. Properties Switch

Special Controls

The set of *Special Controls* elements is composed of types Trace, Waiting Symbol Flower and Text Editor.

The configurations and informations about **User Inputs when Editing an Element e User Interface when Editing the Properties** remain the same presented to the objects of the Basic group.

Trace

This element allows to integrate a trace graph within the visualization for permanent monitoring or displaying of variables. The trace to be displayed is specified in the element properties. Additional control elements are required to control the trace functionalities, which have to be programmed as well. Either manually, as described below or with help of the **Trace Wizard**.

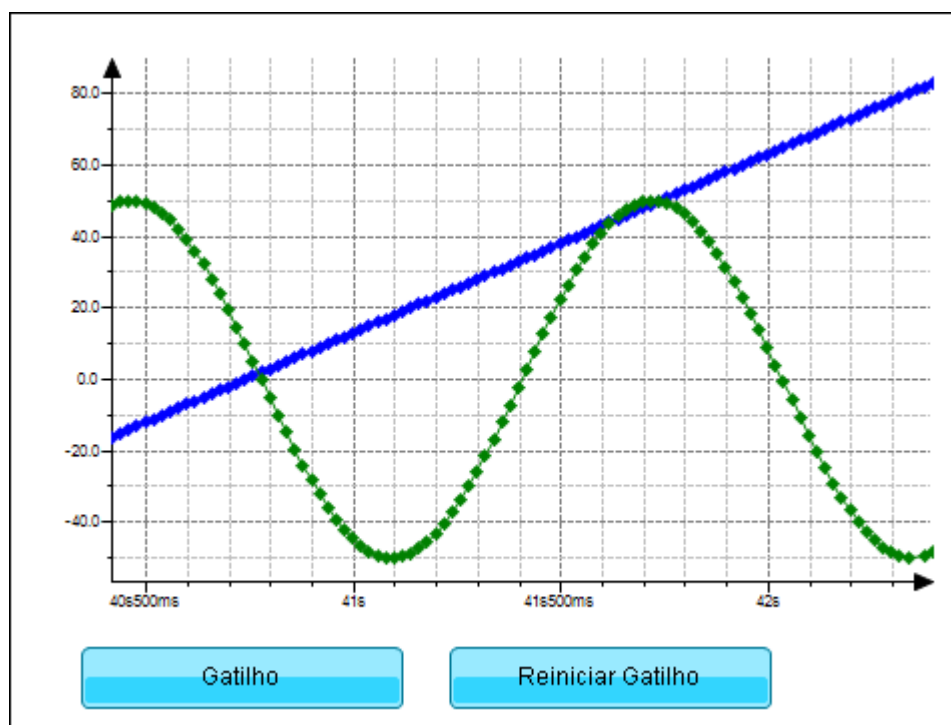


Figure 8-201. Visualization Trace in Run

Properties of Trace

O element *Trace* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Access Rights**

Trace

Property	Value Description
Application	Specify here the application under which the trace variables are implemented. Default is the application under which the visualization is placed.
Trace	With a click in its (unselected) value field, the dialog Trace Configuration, Record Settings opens for configuring the trace graph.
Show cursor	If this option is activated, the values of the trigger and the variable values the cursor points to are displayed as a tooltip.
Overwrite existing trace on PLC	If a trace with identical name is on the PLC, it will be overwritten during a download with the configuration done here.

Table 8-110. Properties Trace

Control Variable

Property	Value Description
Reset trigger	Enter a Boolean variable. If the variable gets TRUE, the trigger is reset.
Start trace	Enter a Boolean variable. If the variable gets TRUE, the trace is started.
Parar o trace	Enter a Boolean variable. If the variable gets TRUE, the trace is stopped.
Save trace to a file - Save trace - File name	If you want to save the current trace to a file, then enter a Boolean variable in the value field of "Save trace" and a STRING variable in the value field of "File name". When the "Save trace" variable gets TRUE, the configuration is saved with the specified file name.
Load trace from a file - Load trace - File name	If you want to load a configuration, then enter a Boolean variable in the value field of "Load trace" and a STRING variable in the value field of "File name". When the "Load trace" variable gets TRUE, the configuration is loaded with the specified file name.

Table 8-111. Properties Control Variable

Effects of Show Cursor

If the option *ShowCursor* in the element properties of the visual trace element is activated, you are able to get the values of the trigger within a tooltip and the traced variables are displayed at a position marked by the position of the cursor in the visual trace element..

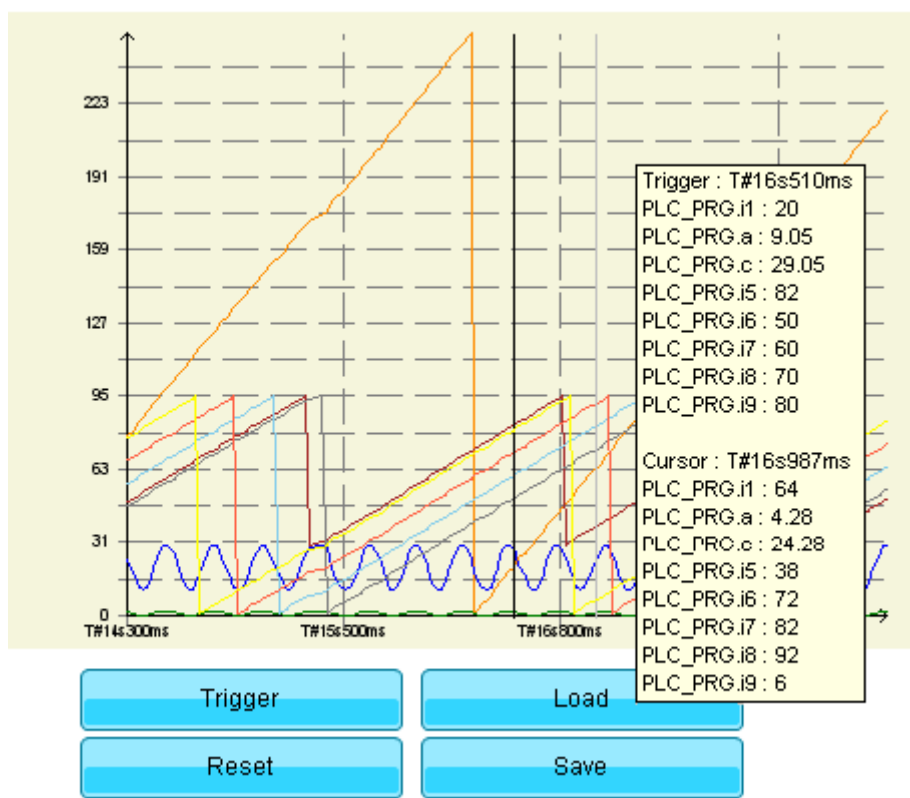


Figure 8-202. Effect of property ShowCursor

Trace Configuration

Trace Configuration, Record Settings

O dialog *Condições de Trace*, *Condições de Registro* abre com clique no Object trace de Visualization em *Visualizar > Properties do Elemento > Trace*. Esse dialog é quase idêntico ao **Record Settings** do **Trace Editor**.

Trigger Basics

Basicamente, existem as seguintes maneiras de controlar o registro das variáveis trace:

- Iniciar e parar o Registro.
- Conectar o Iniciar e Parar com uma condição de registro.
- Disparar um gatilho.

Set and Modify the Record (Trigger) Settings

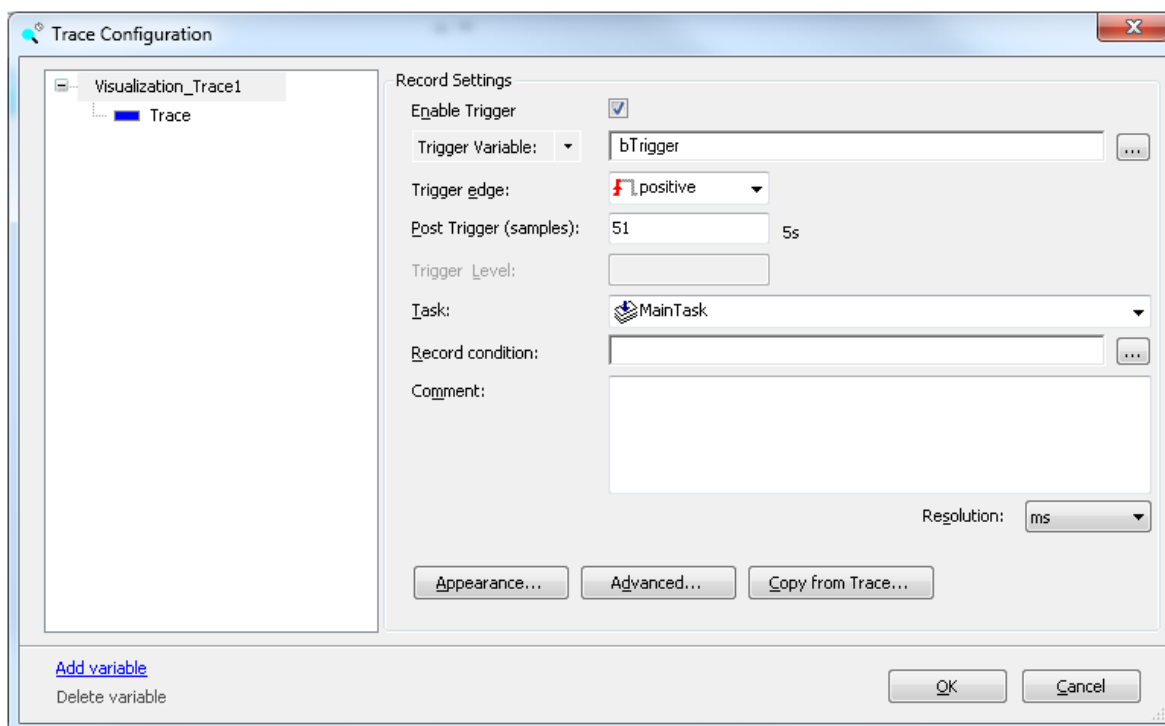




Figure 8-203. Dialog Trace Configuration, Record Settings

Enable Trigger: tick the checkbox for enabling the trigger system according to the settings. If the trigger system is disabled, the trace is free-running.

Trigger Variable: specify which signal will be used as trigger by entering the name (and path) of the signal.

A valid trigger signal is an IEC-variable, a property, a reference, a pointer, an array element of the application or an expression. Allowed types are all IEC basic types except STRING, WSTRING or ARRAY. Enumerations are also allowed, whose basic type is not a STRING, a WSTRING or an ARRAY. The content of a pointer is not a valid signal. The input assistant can be used in order to get a valid entry.

If you want to use a device parameter as trigger, click  and select *Trigger Parameter*. Open the input assistant with  and select *Traceable parameters*. Under *Elements*, the parameters available in the system are listed.

Trigger edge:

- *Positive*: The trigger fires on the rising edge of the Boolean trigger variable or as soon as the value defined by *Trigger Level* for an analogue trigger variable is reached by an ascending run.
- *Negative*: The trigger fires on a the falling edge of the Boolean trigger variable or as soon as the value defined by *Trigger Level* for an analogue trigger variable is reached by a descending run.
- *Both*: The trigger fires both on falling as well as on rising edge of the trigger variable or when the set *Trigger Level* is reached.

Post Trigger: enter here a number of records per trace signal which are recorded after the trigger is fired. Default value is 50, range is from 0 up to $(2^{32}-1)$.

Trigger Level: enter a literal that defines the level at which point the trigger gets fired.

Task: select one from the list of the available tasks where the capturing of the input signals takes place.

Record condition: enter here a variable of type BOOL or of a bit-access. The content of a pointer is not a valid entry. Properties are supported as well.

Comment: enter here an comment text maybe about the current record.

Resolution: set the resolution of the trace time stamp to ms or μ s. For each captured signal, pairs of value and time stamp are stored and transmitted to the programming system. The transmitted time stamps are relative and refer to the start of the tracing. If a task cycle time of less than 1 ms is given, a time stamp with resolution in μ s is recommended.

Appearance...: opens **Edit Appearance**

Advanced...: opens **Advanced Trace Settings**

Copy from Trace...: opens **Copy Settings From Trace Instance**

Trace Configuration, Variable Setting

The *Trace Configuration, Variable Settings* opens when a trace variable is selected in the trace tree. This dialog is almost identical to Trace configuration, **Variable Settings** of the **Trace Editor**.

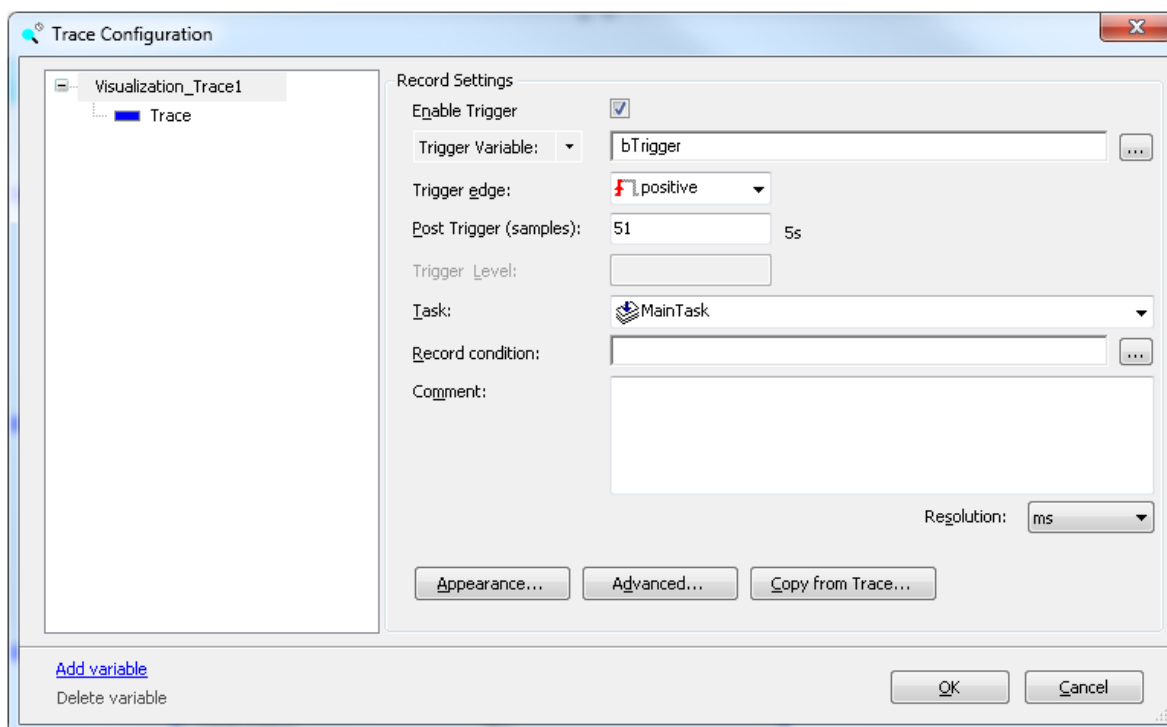


Figure 8-204. Trace Configuration Dialog with Variable Settings

All trace variables are displayed in the left part of the window by a tree structure and the top node is titled with the trace name.

Add and Delete a Trace Variable

Add Variable: with click on it, an anonymous entry appears in the trace tree and in the right part of the dialog the settings of the new variable are ready for configuration.


Delete Variable: with click on it, the selected variable will be deleted with the associated configuration.


The trace tree will be updated immediately.

Set and Modify the Variable Settings

If you want to do the variable settings, select the desired variable in the trace tree and the current settings are displayed in the right part of the trace configuration window. To modify the variable settings later, just select the variable entry in the trace tree and use the *Variable Settings* dialog again.

Variable: Specify which signal will be traced by entering the name (path) of the signal.

A valid signal is a IEC-variable, a property, a reference, the contents of a pointer or an array element of the application. Allowed types are all IEC basic types except STRING, WSTRING or ARRAY. Enumerations are also allowed, whose basic type is not a STRING, a WSTRING or an ARRAY. The input assistant  can be used in order to get a valid entry.

Graph color: choose a color from the given color selection list in which the trace curve for the variable should be displayed. With click on , a color selection dialog opens.

Line type: specify the way samples will be connected in the graph. It is recommended to use *Line* for big volumes of data:

- *Line:* they are connected to a line (default).


- *Step*: they are connected in shape of a staircase. Thus, a horizontal line to the time stamp of the next sample followed by a vertical line to the value of the next sample.
- *None*: they are not connected.

Point type: specify how the values themselves will be drawn in the graph:

- *Dot*: they are drawn as dots.
- *Cross*: they are drawn as crosses.
- *None*: they are not displayed (default).


Activate Minimum Warning: tick the checkbox to display the trace graph in the color Warning minimum color as soon as the variable underruns the value defined in Critical lower limit.

Critical lower limit: enter a value that indicates the lower critical point of the trace variable. If the value of the trace variable underruns it and Activate minimum warning is enabled, the values of the curve changes to the warning color.

Warning minimum color: choose a warning color for underrun from the selection list. With click on , a color selection dialog opens.

Activate Maximum Warning: tick the checkmark to display the trace graph in Warning maximum color as soon as the variable exceeds the value defined in Critical upper limit.

Critical upper limit: enter a value that indicates the upper critical point of the trace variable. If the value of the variable exceeds it and Activate maximum warning is enabled, the values of the curve changes to the warning color.

Warning maximum color: choose a warning color value for exceeding from the selection list. With click on , a color selection dialog opens.

Advanced Trace Settings

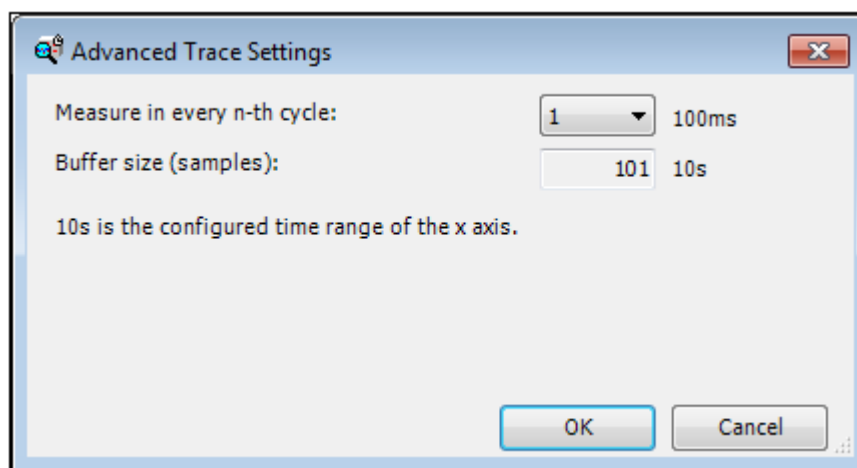


Figure 8-205. Advanced Trace Settings

In this dialog you can configure the size of the trace buffer in the runtime system for the trace element. The number of the measuring points is computed by the values of:

- the cycle time of the task the trace runs in.
- Measure in every n-th cycle
- the length of the x-axis as it is configured in the Appearance Settings: **Edit Appearance**

Buffer size (samples): here, the result is displayed.

Edit Appearance

The *Edit Appearance* dialog opens on command *Appearance...* in dialog **Trace Configuration, Record Settings**.

The settings can be managed as follows:

- *Reset*: the appearance is reset to its default value.
- *Use as default*: the current configuration of the appearance is stored as default. These settings are used if a new trace is configured or a trace variable is added.

X- Axis

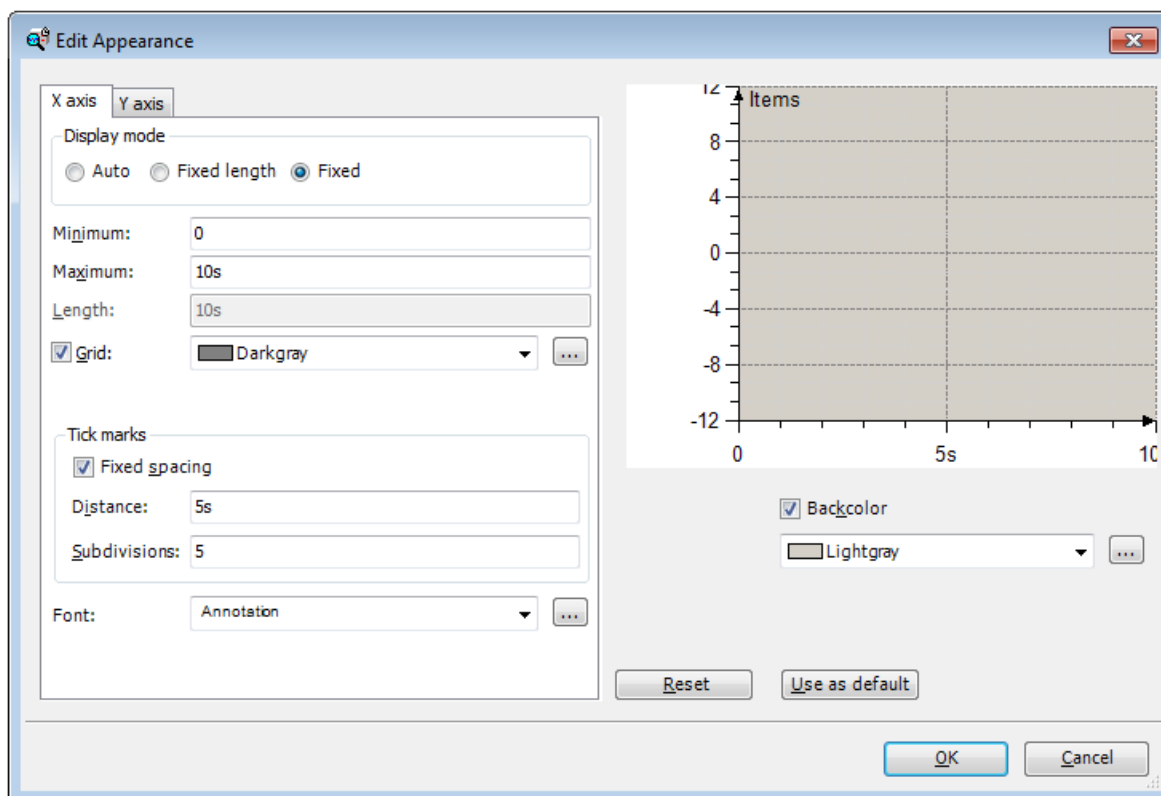


Figure 8-206. X-axis

Here, select a display mode:

- *Auto*: the time axis is scaled automatically according to the contents of the dialog **Advanced Trace Settings**.
- *Fixed length*: the displayed interval of the time axis has a fixed length, that has to be defined in entry *Length*. The scale is also adjusted to the length and the graph is automatically scrolled into a visible range. So, always a time interval with the configured length and the associated latest data is shown in the diagram. But only as many values as were recorded. And as soon as new data are recorded, the display scrolls with.
- *Fixed*: the displayed interval of the X-axis is defined statically by minimum and maximum value.

Minimum: enter here the minimum value to be displayed on the time axis.

Maximum: enter here the maximum value to be displayed on the time axis.

Length: set the length of the displayed interval of the time axis.

Grid: tick the checkbox if you want to display a grid in color. For that, select a color from the drop-down list next to it. With click on [...] a color selection dialog opens.

Tick marks

- *Fixed spacing*: tick the checkbox if you want the axis is scaled in certain distances.
- *Distance*: enter here a positive distance adjusted to the limits of the time axis X. Default distance is 1s.
- *Subdivision*: enter here a reasonable number of subdivisions for each distance.

Font: select a font from the selection list, which is then used in the trace diagram. With click on [...] a font selection dialog opens.

Y-Axis

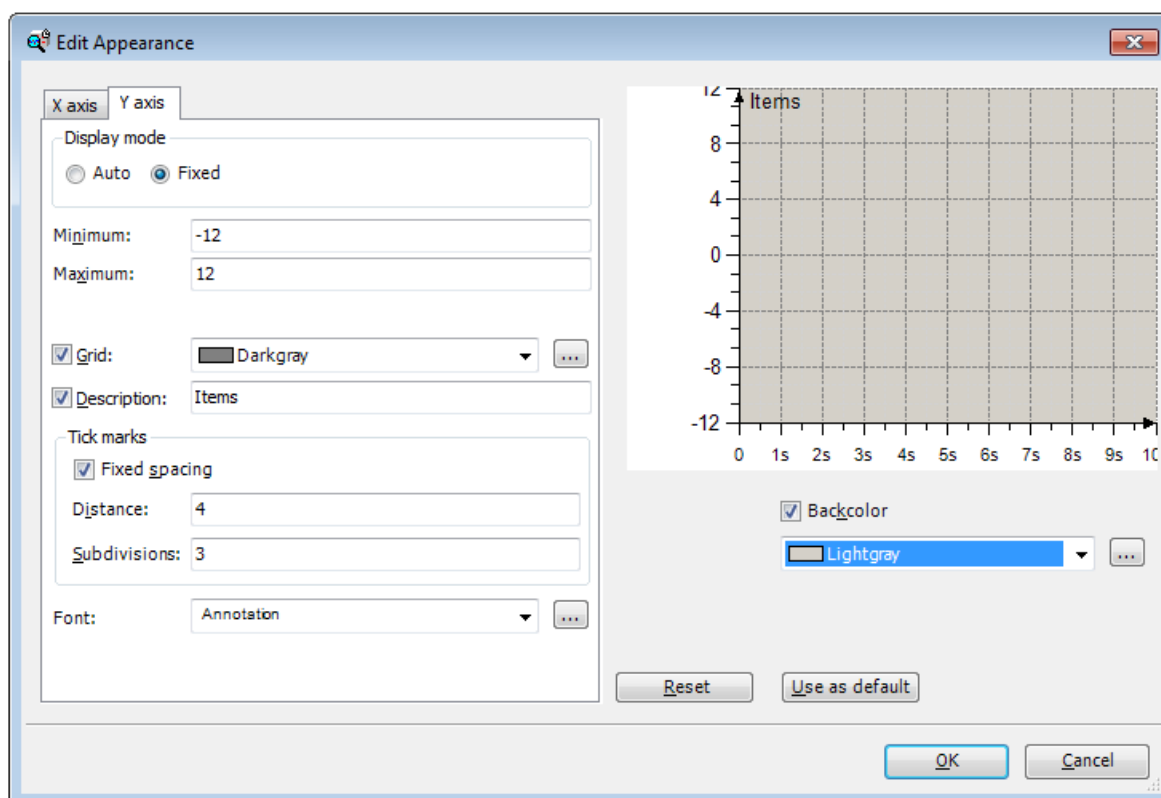


Figure 8-207. Y-Axis

Select here a display mode:

- *Auto*: the Y-axis is automatically scaled according to the captured values. No further entries have to be set.
- *Fixed*: the displayed interval of the Y-axis is fix defined by *Minimum* and *Maximum*.

Minimum: enter here the minimum value to be displayed on the Y-axis.

Maximum: enter here the maximum value to be displayed on the Y-axis.

Grid: tick the checkbox if a grid will be displayed in color. For that, select a color from the drop-down list next to it.

Description: tick the checkbox in order to label the Y-axis with the text entered in the field next to it.

Font: this button opens the standard dialog for defining the font for the trace display.

Tick marks:

- *Fixed spacing*: tick the checkbox in order to scale the Y-axis in certain distances.
- *Distance*: enter here a positive distance adjusted to the limits of the Y-axis and its unit. Default is 1.
- *Subdivision*: enter a reasonable number of subdivisions for each distance.

Font: select a font, which is used in the trace diagram.

Coordinate system:

BackColor: tick the checkbox to display the diagram with the background color given from the color selection list. With click on a color selection dialog opens.

Copy Settings From Trace Instance

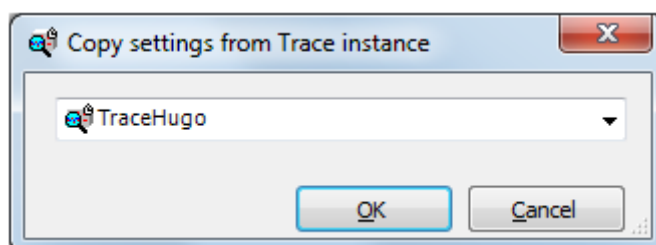


Figure 8-208. Copy settings from Trace instance

Escolha um Object trace da lista para copiar sua configuração para o dialog *Configuração Trace* sendo editado.

Trace Wizard

This dialog opens with command *Insert elements for controlling trace...* located in the context menu of the Visualization Editor or in menu *Visualization*. It is only available if a trace element is selected.

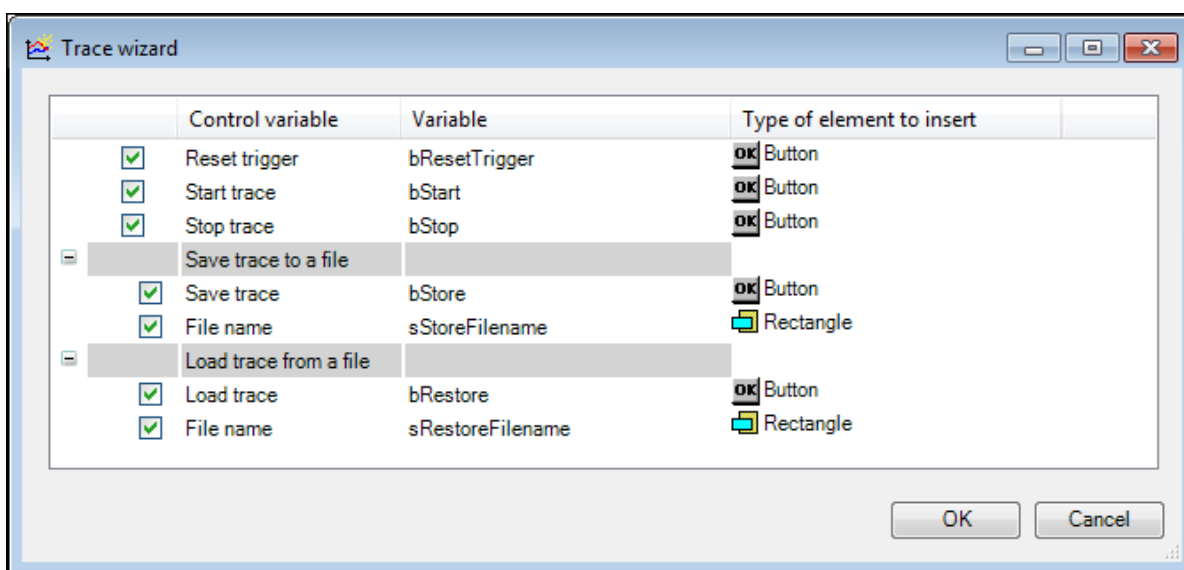


Figure 8-209. Trace Wizard

Control variables: in this column the control variables of a trace element are listed. This list corresponds to the control variables available in the *Properties* view of the trace editor. Tick the checkbox next to the control variable if a control element for this variable is to be created in the Visualization Editor.

Variable: in this column the names of the variables assigned to the trace control variables are listed. This list corresponds to the assignments done in the *Properties* view of the trace editor. If so far no assignment has been done there, a preassignment with standard variable names is proposed here. If you want to change a name, open an inline-editor by double-click on the name or by single click on a selected name field.

Type of element to insert: in this column the type of control elements are listed. If the checkbox is ticked, the element listed here is or will be inserted in the visualization. If you want another type of element to be inserted, open the selection list by double-click on the element or by single click on a selected element and select one. The selection list contains the elements which make good use.

Examples of Trace Configuration

How to Configure a Minimum Visualization Trace Element

1. Declare project variables for control (start and end trace) in IEC-code. Implement a trace variable.
2. Set property *Trace* of the visualization element. A minimum trace contains one trace variable.
3. Assign project variables under *Control variables*, at least for *Start trace* and *Stop trace*
4. Implement the code for the trace variables

In this example, a Visualization Trace element is connected over project variables with input buttons to control the trace and the value curve of the IEC-variable iHugo is displayed.

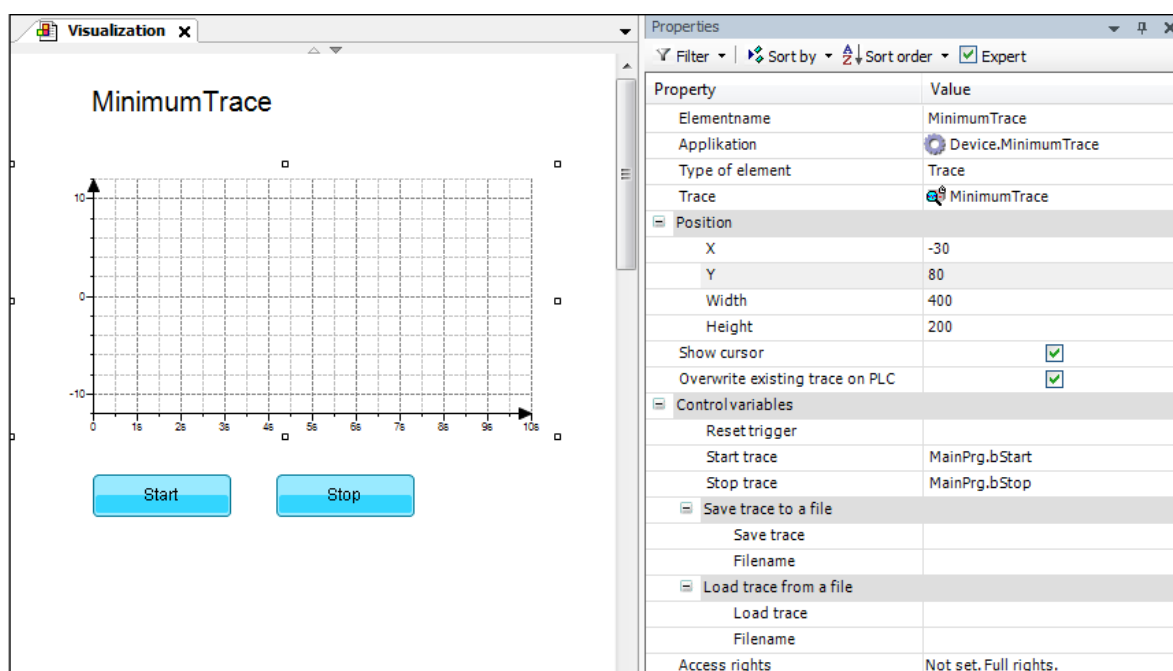


Figure 8-210. Visualization MinimumTrace

Step 1: IEC Code

```
PROGRAM MainPrg
VAR
```



```

END_VAR
VAR_INPUT
    iHugo: INT;
    bStart: BOOL;
    bStop: BOOL;
END_VAR

iHugo := iHugo + 1;

```

Step 2: Trace Configuration Handled in Property Trace

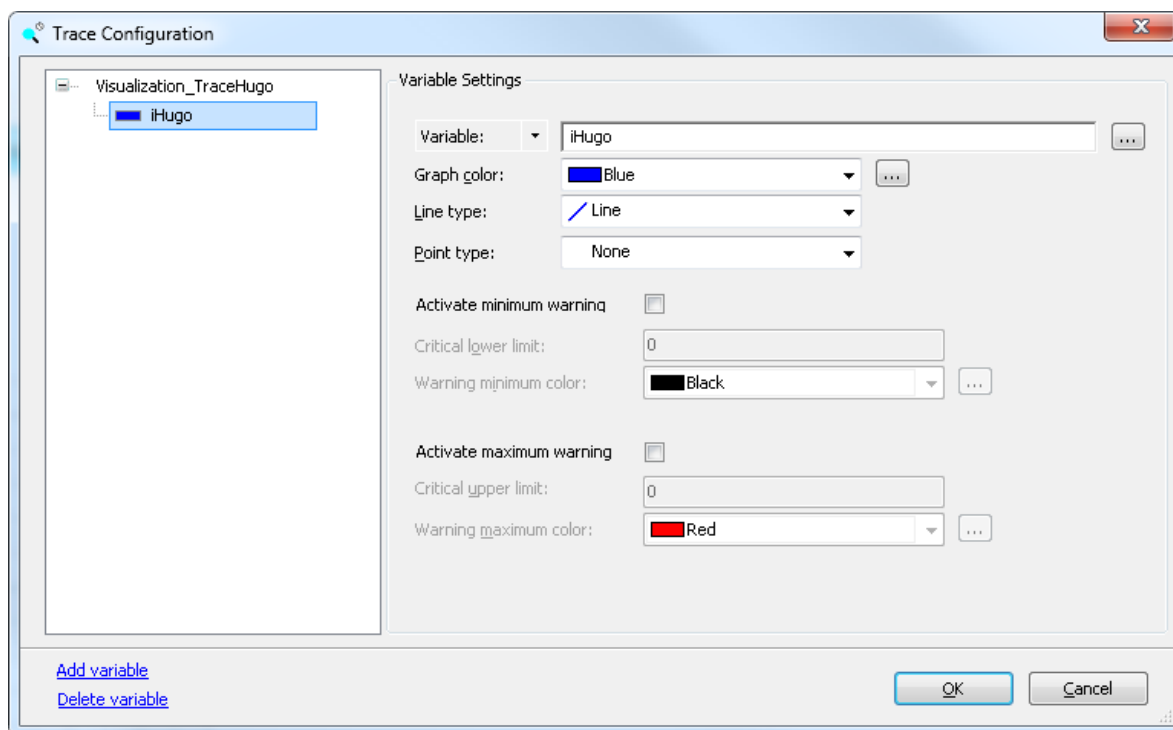


Figure 8-211. Trace Configuration of MinimumTrace

Step 3: Settings in Property Control Variable of Element MinimumTrace

Property	Value Description
Start trace	MainPrg.bStart
Stop trace	MainPrg.bStop

Table 8-112. Properties Start and Stop Trace Configuration

Step 4: Implementation of a Trace Control

An easy way for handle the control variables is to add buttons in the visualization and set an input configuration for them. *Start* is for start the trace, *Stop* is for stop it. Therefore, configure the input configuration of the buttons like this:

ST-code for button Stop programmed in its *Properties* under *Input Configuration* > *OnMouseDown* > *Execute ST-Code*

```

MainPrg.bStart := FALSE;
MainPrg.bStop := TRUE;

```

ST-code for button Start programmed in its *Properties* under *Input Configuration > OnMouseDown > Execute ST-Code*

```
MainPrg.bStart := FALSE;
MainPrg.bStop  := TRUE;
```

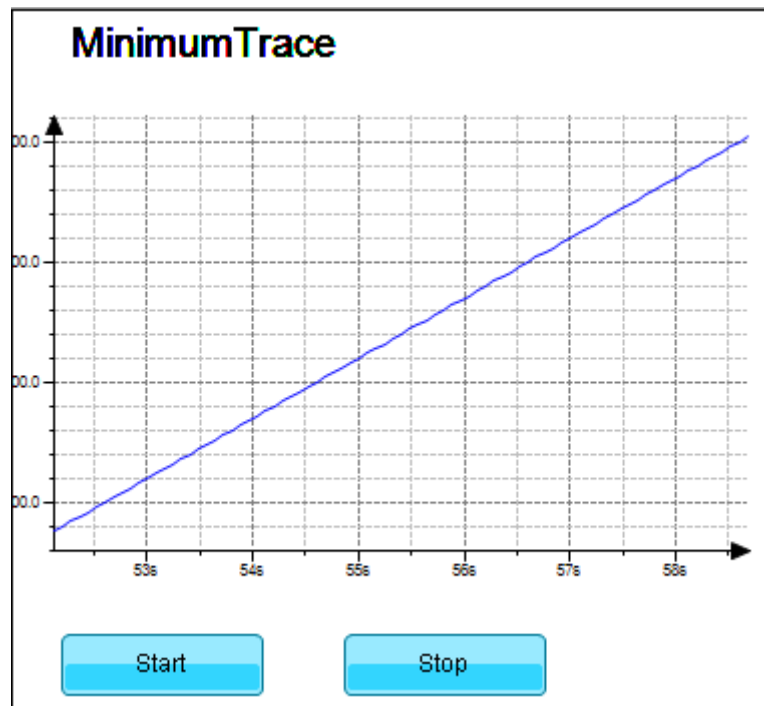


Figure 8-212. MinimumTrace online

How to Configure a Visualization Trace with Trigger

Configure following properties:

1. Declare the needed project variables in IEC-code and implement them for trigger and reset the trace. Minimum therefor is a control variable for trace and one for reset.
2. *Trace*: configure the trigger system in dialog **Trace Configuration, Record Settings**
3. *Control variables*: assign a project variable for *Reset trigger*.
4. Implement the trace control for trigger and reset

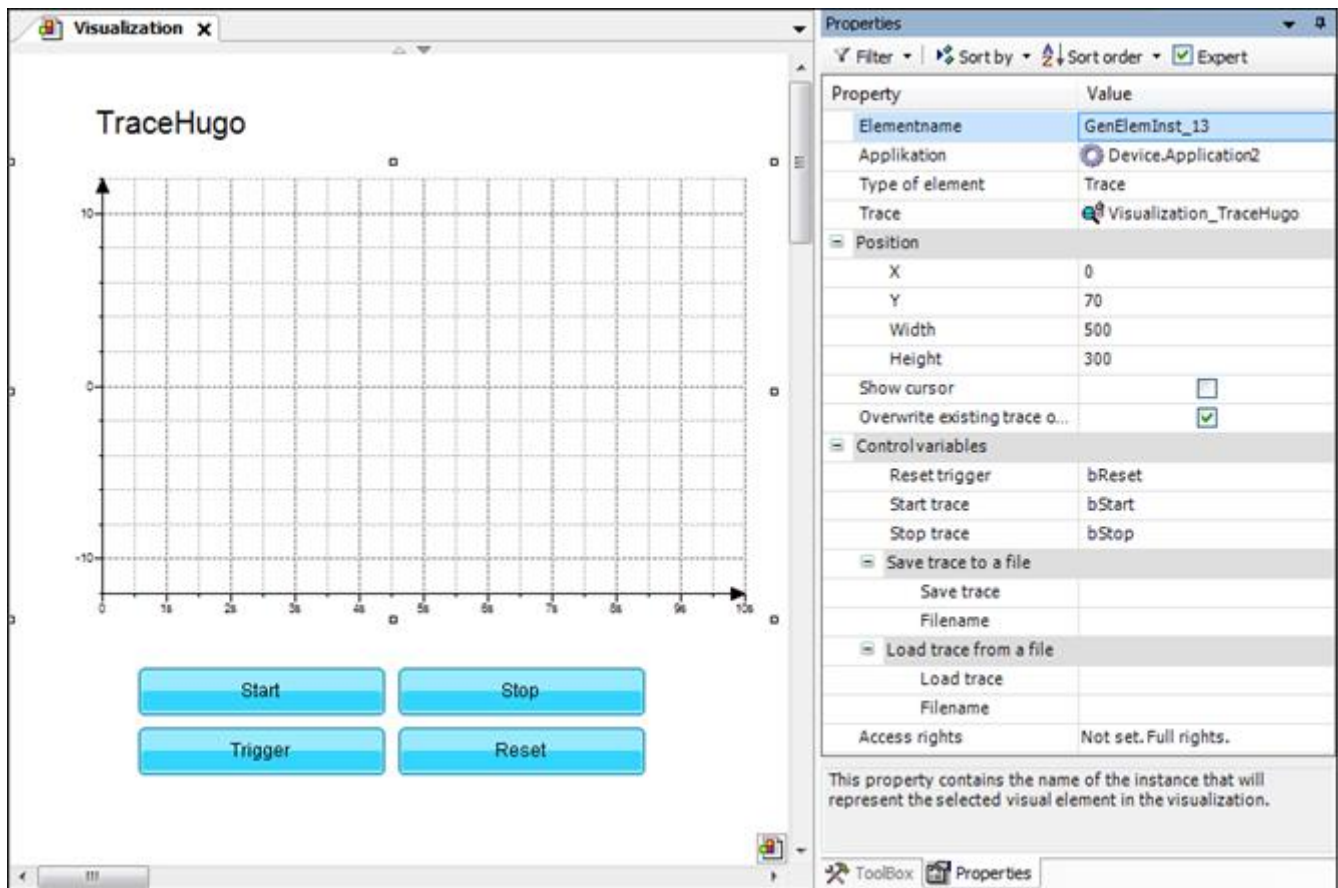


Figure 8-213. Properties of TraceHugo

Step 1: IEC-Code

```

PROGRAM MainPrg
VAR
END_VAR
VAR_INPUT
    iHugo: INT;
    bStart: BOOL;
    bStop: BOOL;
    bTrigger: BOOL;
    bReset: BOOL;
END_VAR

iHugo := iHugo + 1;

```

Step 2: Trace Configuration

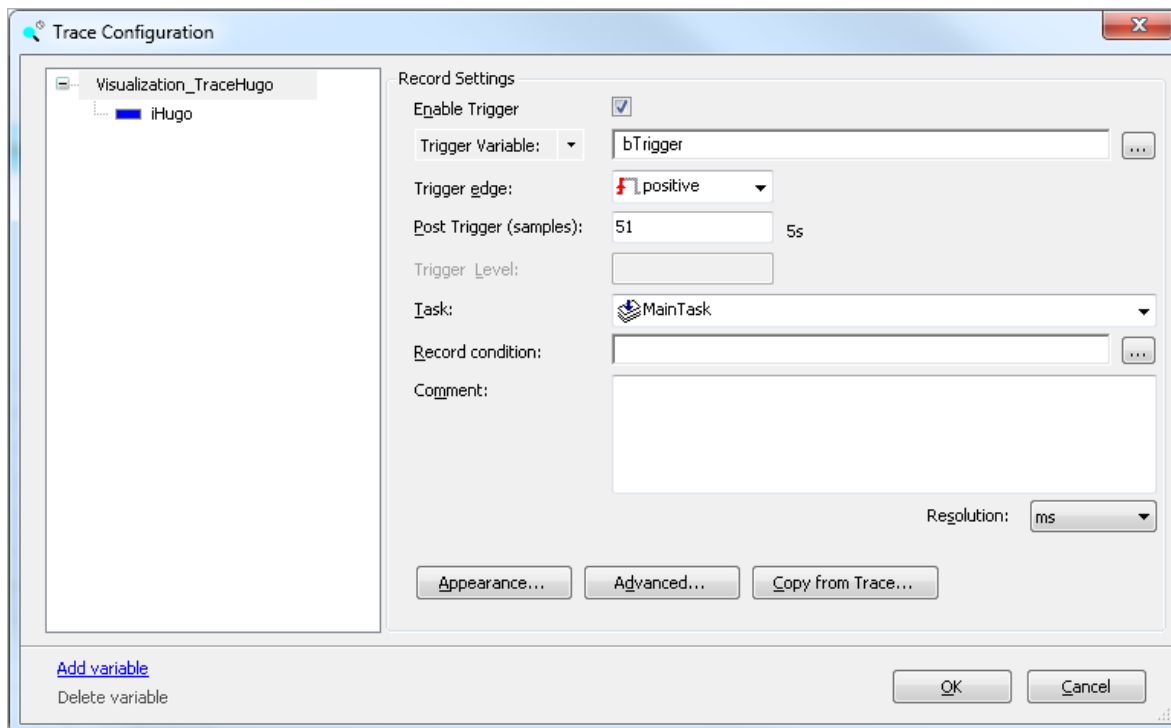


Figure 8-214. Trace Configuration of TraceHugo

Step 3: Settings in property Control Variable of element TraceHugo

Property	Value Description
Reset	MainPrg.bReset

Table 8-113. Property Reset Configuration

Step 4: Implementing a Trace Control

An easy way to handle the control variables is to define buttons with input configuration. Button *Trigger* is for firing the trigger, that means to activate the triggering of the trace by setting the associated trigger variable to TRUE. Button *Reset* is for resetting the trigger. Therefore, configure the input configuration of the buttons like this:

ST-code for button *Trigger* programmed in its *Properties* under *Input Configuration* > *OnMouseDown* > *Execute ST-Code*

```
MainPrg.bReset := FALSE;
MainPrg.bTrigger := TRUE;
```

ST-code for button *Reset* programmed in its *Properties* under *Input Configuration* > *OnMouseDown* > *Execute ST-Code*

```
MainPrg.bTrigger:=FALSE; //reinicia o gatilho
MainPrg.bReset := TRUE; //habilita o reinicio
```

The following picture shows the trace visualization in online mode after the trigger variable has been activated by pressing the button *Trigger*:

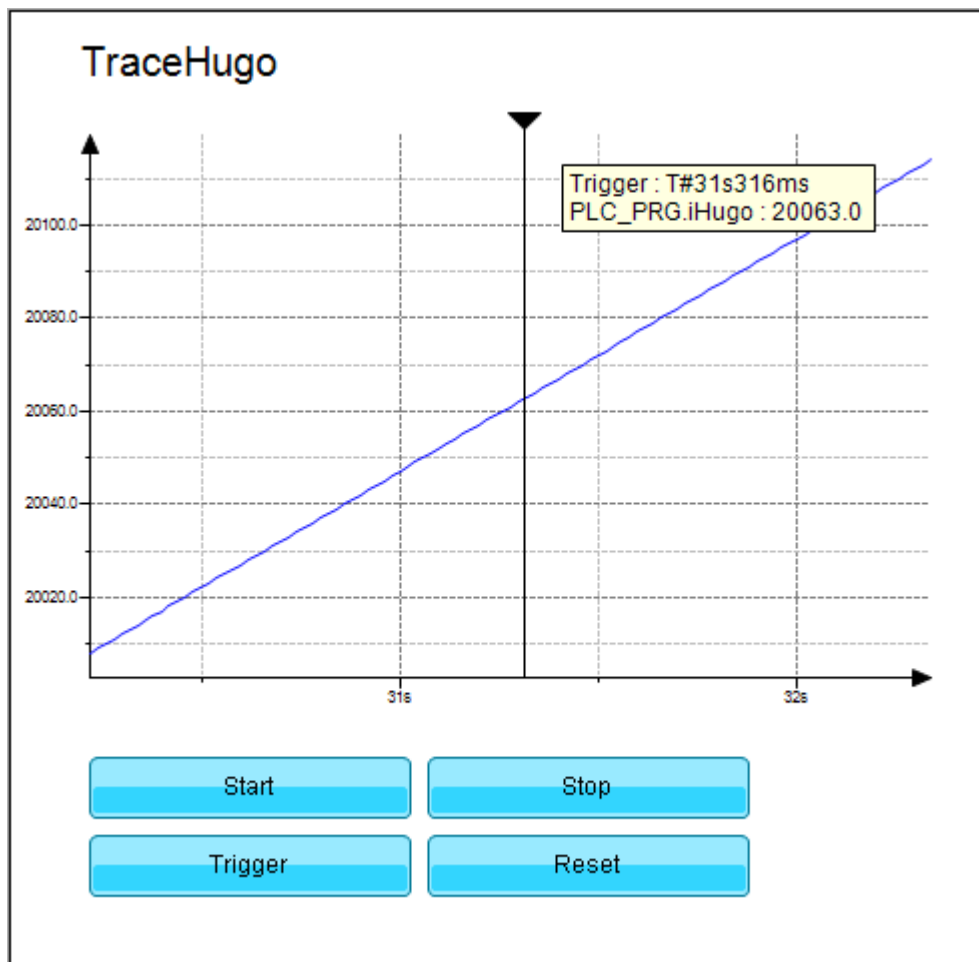


Figure 8-215. Visualization TraceHugo in online mode

The edge of the trigger variable is marked by a black vertical line. Recording of variables continues for the specified trigger level of the whole tracing interval, afterwards the visualization is paused.

Waiting Symbol

Este element animado pode ser usado para sinalizar que o sistema está ocupado ou à espera de dados.



Figure 8-216. Waiting Symbol

Element Waiting Symbol Properties

The element *Waiting Symbol* has the follow settings that are presented in section

Properties:

- **General**
- **Position**
- **Colors**
- **Element Look**
- **State Variables**
- **Access Rights**

But they have some differences, which are listed in sequence:

Colors: this element does not have configuration options of *Alarm state*.

State Variables: this element only have the parameter *Invisible*.

Waiting Symbol

Property	Value Description
Symbol color	Select a color the flower symbol displayed with.
Symbol line width	Enter an integer variable which provides the outline width of the flower symbol in pixels. 0 encodes the same as 1 and set the line width to 1 pixel.

Table 8-114. Properties Waiting Symbol

Text Editor

The visualization element *Text Editor* can be used for displaying and editing the content of text files stored in ASCII or unicode format on the PLC. Several clients can get access on the same text file, only one with write access, the others with read access.

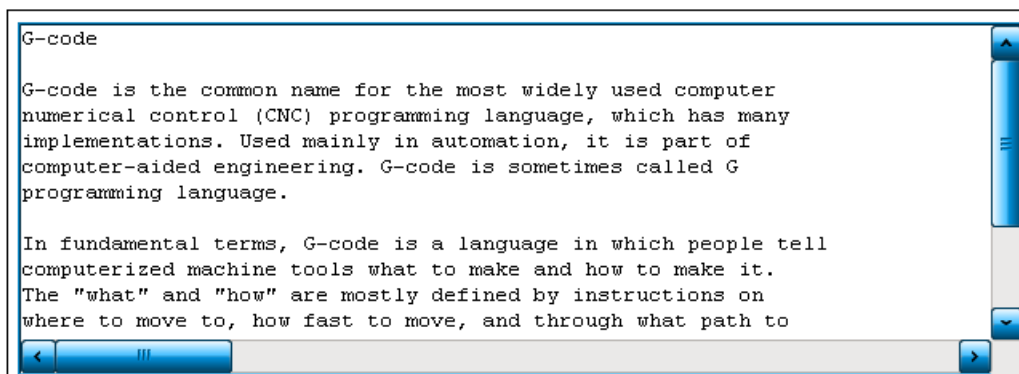


Figure 8-217. Text Editor

Properties of Text Editor

The element *Text Editor* has the follow settings that are presented in section Properties:

- **General**
- **Position**
- **Access Rights**

Font

Property	Value Description
Name	Enter here the name of a font installed on the system for displaying the text in the editor. It has to be a non-proportional font like Courier New
Size	Fix size of the font, for example 12.

Table 8-115. Properties Font

Control Variables

Property	Value Description
Variable for content changed	Atribuir uma variável Booleana. Se o valor da variável recebe TRUE o conteúdo do editor de texto foi alterado.
Variable for read-only/read-write mode	Atribuir uma variável Booleana. Se a variável recebe TRUE, o arquivo tem direitos de acesso somente leitura. Se a variável recebe FALSE, o arquivo tem acesso leitura-escrita.

Table 8-116. Properties Control Variables

File

Property	Value Description
File name	Assign a STRING variable containing the complete file name and path if required. Default path is the installation directory, for example ".\GatewayPLC"
Open	Assign a Boolean variable for controlling the opening of the file specified in "File name". If the variable gets TRUE, the file opens.
Close	Assign a Boolean variable for controlling the closing of the file. If the variable gets TRUE, the file closes.
Save	Assign a Boolean variable for controlling the saving of the file. If the variable gets TRUE, the file is saved.
New	Assign a Boolean variable for creating a new file. If the variable gets TRUE, a new file with the name specified in "File name" opens.

Table 8-117. Properties File

Edit

Property	Value Description
Search for	Atribuir uma variável STRING, contendo a string da pesquisa.
Find	Atribuir uma variável Booleana para controlar a procura da string de pesquisa no arquivo. Se a variável recebe TRUE, a pesquisa inicia.
Find next	Atribuir uma variável Booleana para controlar a procura para a próxima string de pesquisa no arquivo. Se a variável recebe TRUE, os estados de pesquisa na posição atual.

Table 8-118. Properties Edit

Caret Position

Property	Value Description
Line	Assign an integer variable containing the current line number if "Trigger set" is FALSE.
Column	Assign an integer variable containing the current column number if "Trigger set" is FALSE.
Position	Assign an integer variable containing the current position of the caret in

	consecutive numbering if "Trigger set" is FALSE.
Trigger set	Assign a Boolean variable for controlling the caret position. If the variable is FALSE, the variables in "Line", "Column" and "Position" containing the current value. If the variable gets TRUE, the caret is positioned at the position specified in "Line" and "Column".

Table 8-119. Properties Caret Position

Selection

Property	Value Description
Start position	Assign an integer variable containing the start position of the text selection in consecutive numbering.
End position	Assign an integer variable containing the end position of the text selection in consecutive numbering.
Start line number	Assign an integer variable containing the start line number of the text selection.
Start	Assign an integer variable containing the column index the text selection started.
End line number	Assign an integer variable containing the end line number of the text selection.
Índice da coluna final	Assign an integer variable containing the end column index of the text selection.
Line to select	Assign an integer variable containing the line of the text selection.
Trigger select	Assign a Boolean variable for controlling the selecting of the text. If the variable value gets TRUE, the selection is positioned like it is specified in "Line to select".

Table 8-120. Properties Selection

Error Handling

Property	Value Description
Variable for error code	Assign an integer variable containing the error code when an error occurs. The codes are defined in GVL_ErrorCodes part of library VisuElemTextEditor. To get the error text to the error code the following function is provided by the library: VisuFctTextEditorGetErrorText(). See Example "Error messages."

Table 8-121. Properties Error Handling

TextEditor

Property	Value Description
Maximum line length	Input of an integer value that indicates the maximum length of a line.
Editor mode	Select one: - Read-only: read-only access right on the file. In this mode the editor is highlighted in light grey when the visualization is running. - Read/write: read-write access on the file.

Table 8-122. Properties TextEditor

New Files

Property	Value Description
Encoding	If a new file is to be created, define the character coding: - ASCII - Unicode (Little endian) - Unicode (Big endian)
New line character sequence	If a new file is to be created, define here the newline character due to the operating system:

	- CR/LF: normally used in Windows systems - LF: normally used in Unix systems If a existing file is opened, the new line character in the opening file is identified and used automatically.
--	--

Table 8-123. Properties New File

User Inputs when Navigating in the Online Text Editor

The text editor provides user inputs in order to navigate in it during the visualization is in running mode. All of them are commonly known.

User input	Description
[Seta para esquerda]	The caret is moved one character to the left. If the current caret position is the beginning of a line it is moved to the end of the previous line if one exists.
[Seta para direita]	The caret is moved one character to the right. If the current caret position is the end of a line it is moved to the beginning of the next line if one exists.
[Seta para cima]	The caret is moved into the previous line.
[Seta para baixo]	The caret is moved into the next line.
[Home]	The caret is moved to the beginning of the current line.
[End]	The caret is moved to the end of the current line.
[Page Up]	The previous page is displayed.
[Page Down]	The next page is displayed.
[Shift] + [Seta para esquerda] [Shift] + [Seta para direita]	Text selection.
[Del] enquanto o texto está selecionado	The selected text is deleted. If nothing is selected, the character right to the caret position is deleted.
[Enter]	A new line is created and the caret is set at the beginning of this new line.
[Ctrl] + [Tab]	A tab character is inserted. Currently a tab is displayed by a single blank.
[Tab]	The focus is moved to the next visualization element

Table 8-124. User Inputs

Text Editor with Control Elements

To get access to the controlling variables the text editor supports, it is necessary to add additional visualization elements to the visualization and connect them with the text editor with control variables.

Control Element Load

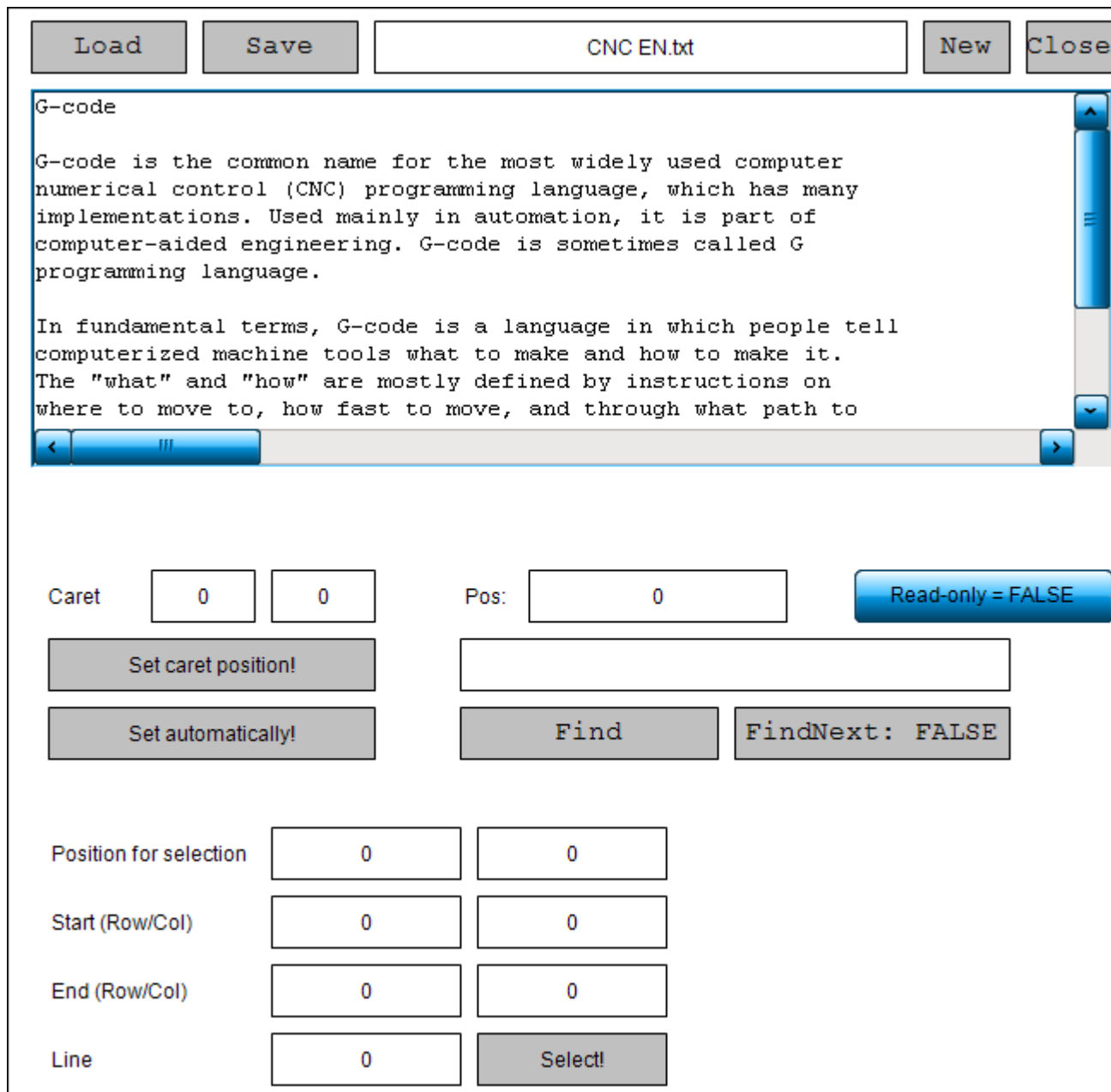


Figure 8-218. Visualization with Text Editor and Several Control Elements

To connect visualization element *Load* with the text editor element do following:

1. Declare the control variables of the element *Load* in IEC-code for example in *MainPrg*:
`bOpen : BOOL;`
2. Declare a variable for "File name" of the text editor in IEC-code for example in *MainPrg*:
`stFileName : STRING := 'MyText.txt'`
3. Add an rectangle element to the visualization and configure it:
Texts > Text : Load
Input configuration > Toggle > Variable : MainPrg.bOpen

4. Add a text editor element to the visualization and configure it:

Control variables > *File* > *File name* : MainPrg.stFileName

Control variables > *File* > *Open* : MainPrg.bOpen

The same can be done with all control variables listed in the properties.

Error Messages

To get out the error text for the error code provided in *Control variables* > *Error handling* > *Variable for error* program the function call *VisuFctTextEditorGetErrorText()* in IEC-Code.

1. Declare the required variables:

```
usiErrorCode : USINT;
stErrorMsg : STRING(255);
```

2. Implement the function call:

```
stErrorMsg := VisuFctTextEditorGetErrorText(usiErrorCode);
```

3. Get out the error text in a visualization for example in a rectangle element. Therefor add a rectangle with following properties:

Texts > *Text* : %s

Text variable > *Text variable* : POU.stErrorMsg

Visualization Libraries

Due to the fact that visualization elements are implemented as function blocks, they are provided via libraries.

When adding a visualization object in a project, a certain set of visualization libraries will be added to the Library Manager in the POU's view. The names as well as the versions of these libraries are defined by the currently used **Project Settings**. The profile also defines in particular which elements coming from the libraries will be available in the **Toolbox** of the visualization editor.

A visualization library always is set up as a special type of *placeholder library*, which effects that the exact version of the library to be used is not resolved until the library gets included in the project. Only then the currently active profile will define which version actually is needed. Notice that this type of libraries is different from the device-specific *placeholder libraries*, where the placeholders get resolved by the device description.

Basic libraries are included by default when adding a visualization object in a standard project. These libraries include further libraries, which are not listed here. Usually you do not have to include the visualization libraries manually or to use them explicitly in your applications::

Basic Libraries

VisuElems: basic set of visualization elements

VisuElemMeter: meter and bar display elements

VisuElemWinControls: table, textfields, scrollbar elements


VisuElemTrace: trace element

VisuInputs: handling of inputs on a visualization

In visualizations associated to a device there might be device-dependent restrictions concerning the configuration possibilities (for example: list of available elements, fonts, colors, image formats, max. number of visualization elements in a visualization, max. number of visualizations below the device) due to the libraries the device supports.

Visualization Manager with Clients

Visualization Manager

If a visualization is inserted in the *device tree* below an application, automatically a *Visualization Manager* object  will be added. It handles common settings for all visualizations assigned to the current application by clients. To open the Visualization Manager object, perform a double-click on the entry in the device tree or select the entry and use the *Edit Object* command from the context menu. The editor opens in a window titled with *Visualization Manager <application path>* containing separate tabs for *Settings*, *Default Hotkeys*, *Visualizations* and *User management*. The settings will be applied immediately after modifications.

Settings

The dialog *Settings* is part of the *Visualization Manager editor* and the settings here are valid for all applications located under the same application as the manager..

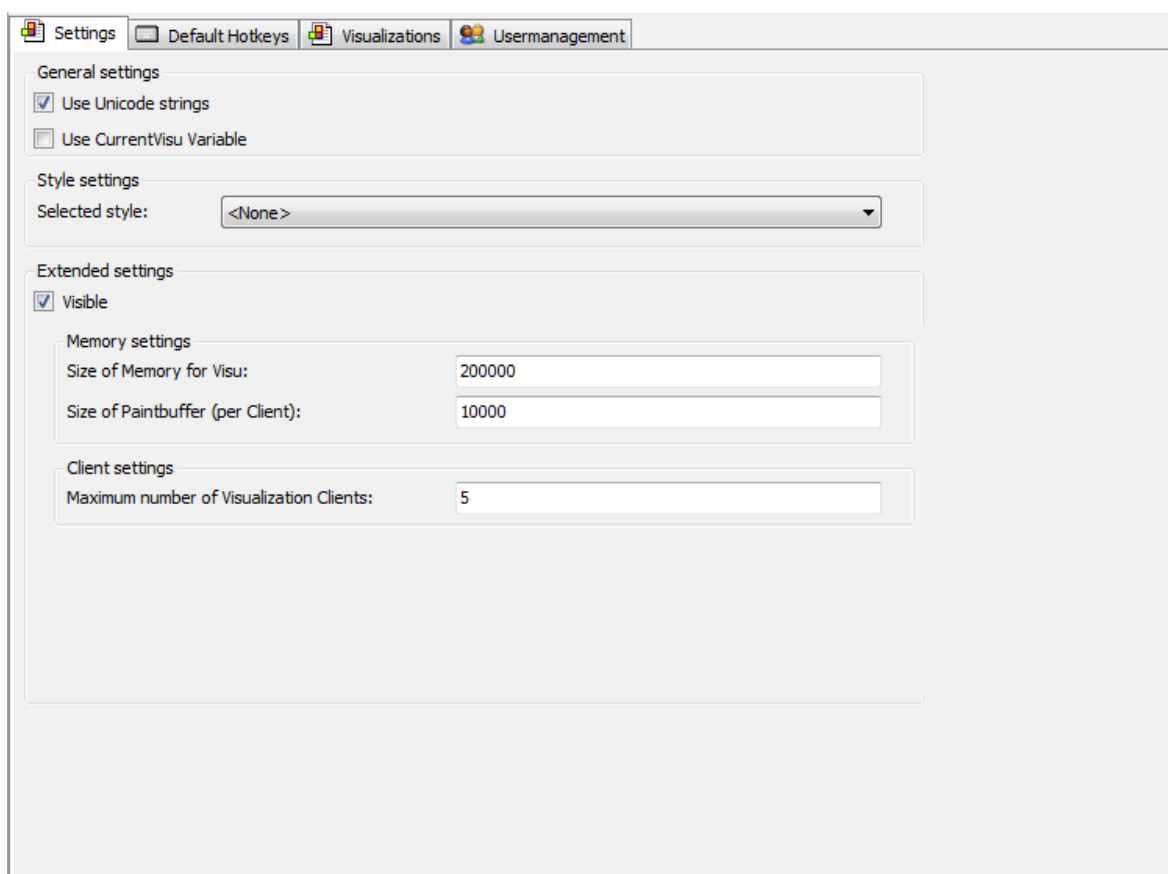


Figure 8-219. Visualization Manager, Settings

General settings:

- *Use Unicode strings* : if this option is activated, all strings used in the visualizations will be processed in Unicode format. **Text and Language in Visualization.**
- *Use CurrentVisu Variable* : if you activate this option, the string variable CurrentVisu, accessible via VisuElems.CurrentVisu, will be taken into account. This variable contains the name of the currently displayed visualization and can be used in the application program to control which visualization currently should be displayed. If a new string is assigned to CurrentVisu, a programmatic visualization switch will be executed. Examples:

- Using a string variable: `VisuElems.CurrentVisu:=strVisuName;`
- Direct assignment of the visualization name: `VisuElems.CurrentVisu:='visu1';`

Extended settings:

- *Visible* : activate this option if you must modify the following settings, which usually is not necessary for standard visualization applications.
- *Memory settings:*
 - *Size of Memory Visu* : memory space in Bytes, allocated for visualization, default: 400000.
 - *Size of Paint buffer (per Client)* : memory buffer in Bytes, allocated for painting actions on the client, default: 50000.

Example

The Visualization uses memory that is initially set at Size of Memory for Visu and is allocated in data area.

Memory usage is based on the following formula according to the example below.

- Size of Memory for Visu: 400,000
- Size of Paintbuffer (per client): 50,000
- The Visualization needs about $1.5 * \text{Size of Paintbuffer (per client)}$ more rounding off.

Portanto:

$\text{Size of Memory for Visu} = 1.5 * \text{Size of Paintbuffer (per client)} + n * \text{Size of Paintbuffer (per client)}$, where n is the number of clients who want to configure.

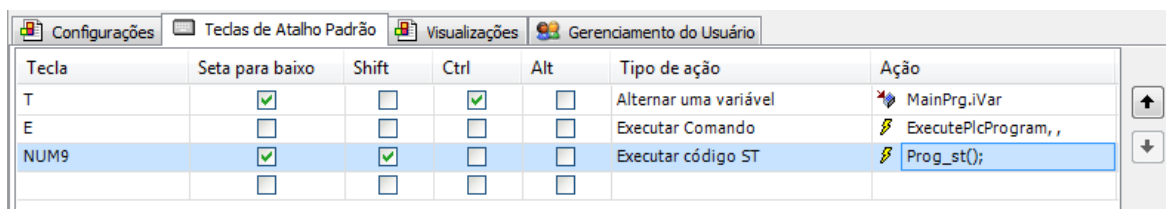
The default value that is set is for 6 clients:

- $\text{Size of Memory for Visu} = 1.5 * 50,000 + 6 * 50,000$
- $\text{Size of Memory for Visu} = 375,000 + \text{rounding off}$
- $\text{Size of Memory for Visu} = 400,000$

Client settings

- *Maximum number of Visualization Clients:* the number of clients is limited to Maximum number of visualization clients. That is required when a visualization has to run on different clients and some settings should be set client dependent. For this purpose, each client gets an ID and data can be provided client dependent. The ID of the current clients can be get by querying the system variables `CURRENTCLIENTID`. For example `arr[CURRENTCLIENTID].dwColor`. The functionality to support client specific states in a visualization is provided in library `VisuGlobalClientManager`.

Default Hotkeys



Tecla	Seta para baixo	Shift	Ctrl	Alt	Tipo de ação	Ação
T	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Alternar uma variável	MainPrg.iVar
E	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Executar Comando	ExecutePlcProgram, ,
NUM9	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Executar código ST	Prog_st();
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Figure 8-220. Visualization Manager Editor, Default Hotkeys

This tab of the *Visualization Manager* dialog contains the key configuration valid for all visualizations assigned to the application. Shortcuts can be defined which - if supported by the respective device - can be used for user inputs on the visualizations in online mode via the keyboard. This works just like the **Hotkeys Configuration**. Additionally always some standard shortcuts for

navigation within a visualization are supported device-independently (**Keyboard Usage in Online Mode**).

Visualization

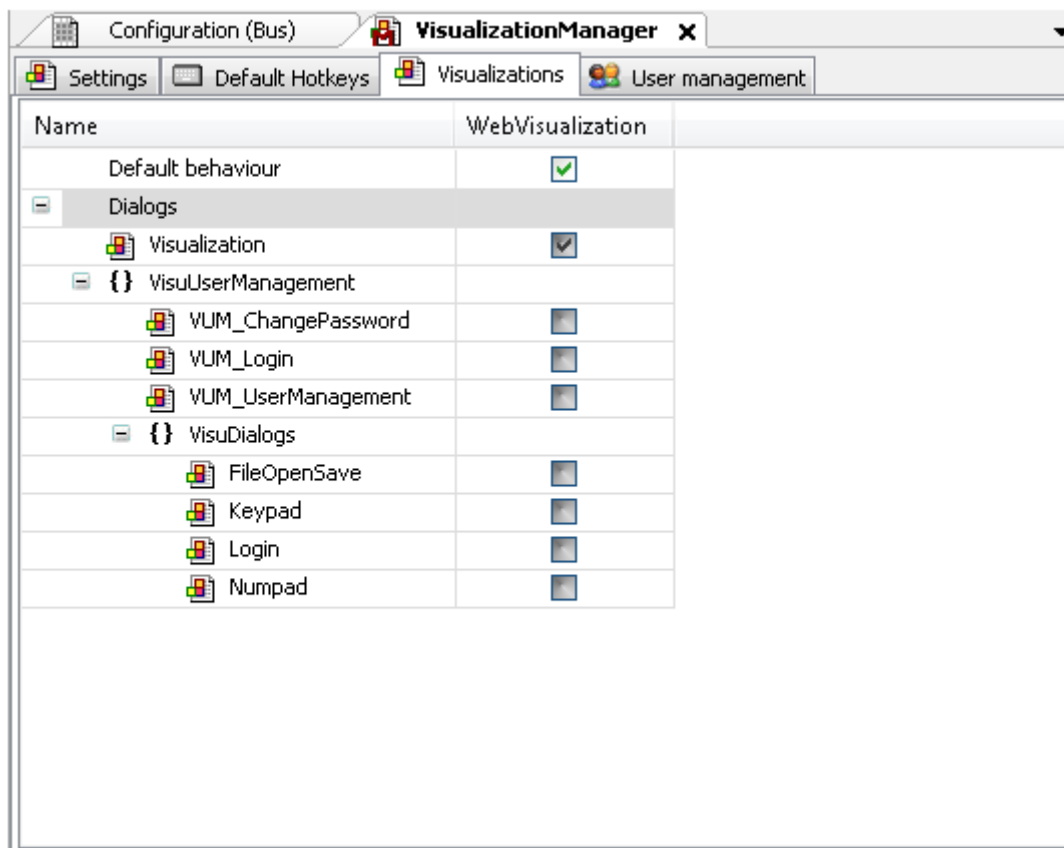


Figure 8-221. Visualization Manager Editor, Visualizations

This tab of the *Visualization Manager* shows the currently available visualizations. By default only those which are located below the current application or are directly referenced by others will automatically get downloaded to the respective target system. Referenced visualizations are located global or are in libraries. If option *Default behavior* is activated, exactly those will be marked in the table with a checkmark in the respective column *WebVisualization*. If you want to add further visualizations to the download package, which are not included by default because they are only used/referenced indirectly (for example via variable by the application), then add the appropriate checkmarks..

User Management

The user management is used for handling the visibility and operability of visualization elements user dependent. Even the switching of visualizations can be configured user dependent. The users can be organized in groups.

Getting Started

First you have to configure users by defining groups and unique user entries. Open tab *User management* in editor *Visualization editor*. If the user management hasn't been configured yet, an assistant provides following dialog for creating one:

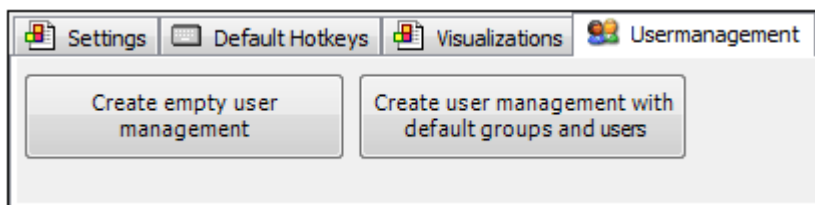


Figure 8-222. Creation of a First User Management

Create empty user management : with using this command, a user management is created with group None. Then the standard view of the user management opens.

Create user management with default groups and users: the user management view opens with the configured groups Admin, Service, Operators and “None”, and the users Admin, Service and Operator. Thus, a viable user management is available.

Programming your visualization

1. Configure your user management by defining **Groups** and **Users**. One user management per system is expected.
2. Integrate the dialogs provided from library VisuUserManagement into your visualization by programming elements with input configuration. A dialog/action programmed with **User Management** runs on an defined event.

Or program your own dialogs.

3. Program the visualization elements by setting the property **Access Rights** in its property view. Then it behaves groups dependent. In **Element List** the elements with limited rights are marked.

If Groups and Users are defined should be instanced a login dialog box but case not instanced the warning messages appear in Generate Code. Example:

The dialog is not instantiated within your application. Is it missing within the visualization manager?

User management in Diagnosis Visualization

Although there are no clients available below the Visualization Manager, see **Diagnosis Visualization**, it can be required to provide a Visualization User Management in the PLC. For example, HMI clients log in a PLC with integrated visualization. In such a case a option is available:

Use visualization user management on the PLC: If the checkbox is marked, then code for the Visualization User Management is loaded into the PLC.

Groups

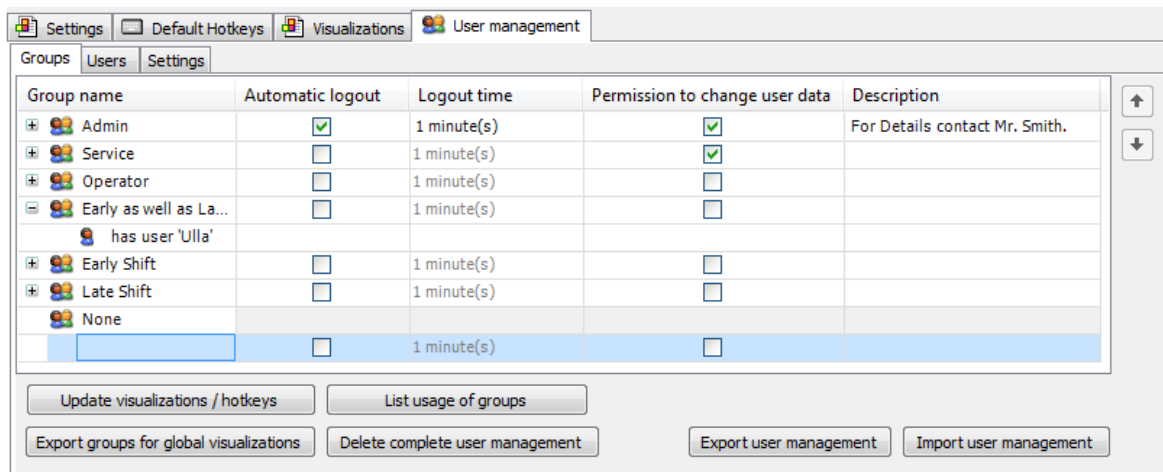


Figure 8-223. User management, Groups

In view *Groups*, all configured groups are listed in hierarchical order.

Group name: if you expand the group, all users assigned to this group are listed in tree view.

Automatic logout: activate the checkbox if you want to define a certain time for logout.



Logout time: set here the time after which you want to be logged out. Use the inline editor to input a number and the selection list to set a unit.

Permission to change user data: activate the checkbox then the group gets the permission to change user data when the visualization is online.

Description : enter here your comments and remarks on the groups. This description is only available in the programming system. It will not be on the PLC.

Add a group: click in the edit line column *Group name* at the bottom of the table to create a new group. Enter an unique name.

Delete a group: mark a group by selecting a field in the table. When you use [Del] this line and, thus, the group is deleted. *None* cannot be deleted.

 : if you want to organize the groups hierarchical in a certain order, click on the arrow-up symbol to move the selected group one line up. With click on the arrow-down symbol, the selected group is moved one line down. The order in the table reflects the hierarchical order. This order can be sent to the PLC and to be available in Online Mode.

Users

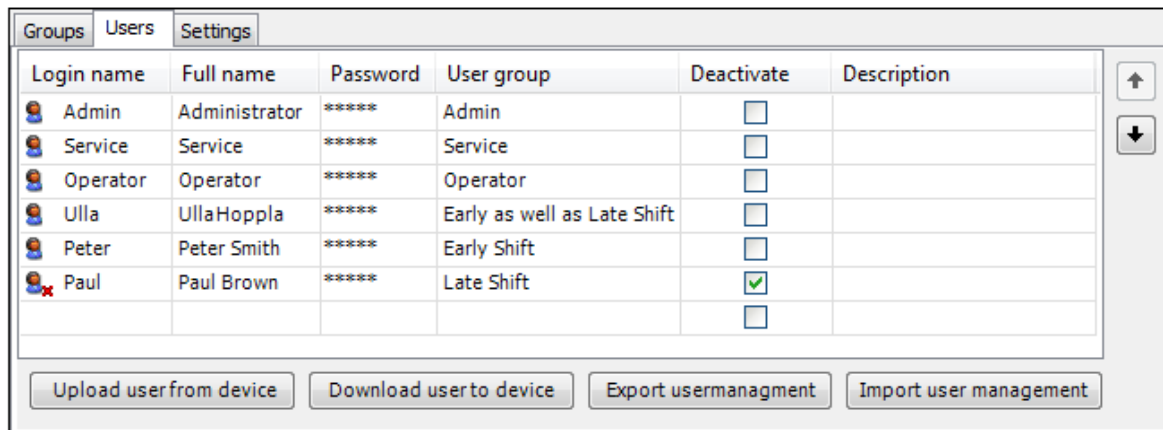


Figure 8-224. Visualization manager, User management, Users

Login name: enter a unique and compilable name.

Full name: enter the full name of the user that can exists multiple times.

Password: enter a password that gets encrypted. Every new user gets its login name as a default password.

User group: enter one user group to which the user will belong.

Deactivate: if the user is to be deactivated, set the checkbox TRUE.

Upload user from device: here, the user management data are uploaded from the PLC in the dialog. If the user management data have already been configured, they will be overwritten.

Download user to device: the data are loaded down to the PLC. If data have already been available, they will be overwritten.

Export user management: a standard dialog opens in order to save a CSV file in any directory with any name. The CSV file contains the user and group data.

Import user management: a standard dialog opens in order to load a CSV file located in any directory with any name. If the format is compatible, it is shown here..

CSV File with User Management Data

The user management data are saved as CSV file in following format.

- User groups:

```
ID;group name;automatic logoff TRUE/FALSE;logoff time;unit logoff
time;permission to change user date TRUE/FALSE
```

- User:

```
login name;full name;password encrypt TRUE/FALSE;password;group ID;user
deactivated TRUE/FALSE
```

Using this format you are able to edit the file with any tool. If *password encrypt* is set to FALSE, you can enter a unencrypted default password like it is done with user *Hugo*. After an import it will be immediately encrypted.

```
V1.0.0.1
```

```
Usergroups:
```

```
1;Admin;TRUE;1;Minute;TRUE
```

```

2;Service;FALSE;1;Minute;TRUE
3;Operator;FALSE;1;Minute;FALSE
6;Early as well as Late Shift;FALSE;1;Minute;FALSE
4;Early Shift;FALSE;1;Minute;FALSE
5;Late Shift;FALSE;1;Minute;FALSE
0;None;FALSE;1;Minute;FALSE

```

User:

```

Admin;Administrator;TRUE;2A4409796B6F78797ED4DEF9BCC9A033;1;FALSE
Service;Service;TRUE;C08298D42A35732CFFB7DF43771B7607;2;FALSE
Operator;Operator;TRUE;3D94AB9540B025B07773DE7037F19837;3;FALSE
Ulla ;Ulla Hoppla;TRUE;7DE74B6B911AA7F034FEF4FBF74456D9;6;FALSE
Peter;Peter Smith;TRUE;1A7F795CBC2B84052201D6718C4AB983;4;FALSE
Paul;Paul Brown;TRUE;6C4A18FC48ED5BE2A2C933ED1CC03614;5;TRUE
Hugo;Hugo Green;FALSE;Green;5;FALSE

```

Settings

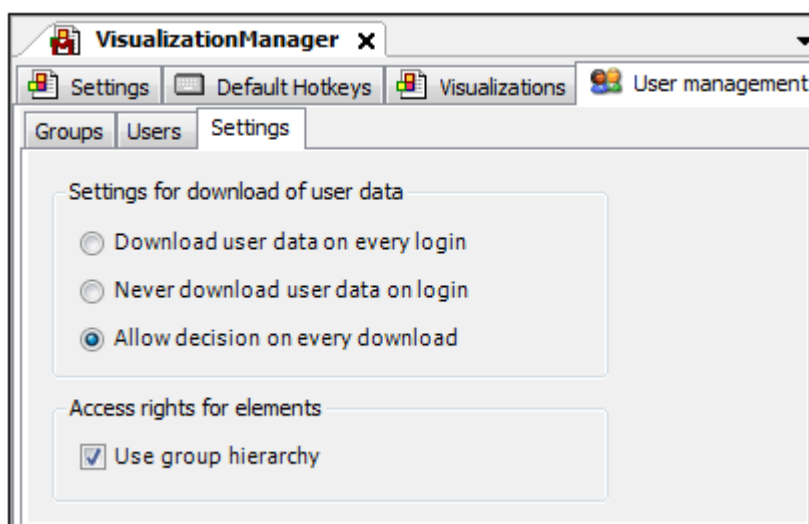


Figure 8-225. Settings

Settings for download of user data: Select a login procedure:

- *Download user data on every login:* this causes that the user management data saved in the programming system are loaded on the PLC on every login and the existing data there will be overwritten.
- *Never download user data on login:* here, the user data are never downloaded even when they have been changed.
- *Allow decision on every download:* this enables a dialog-guided download of the data starting with these warning.

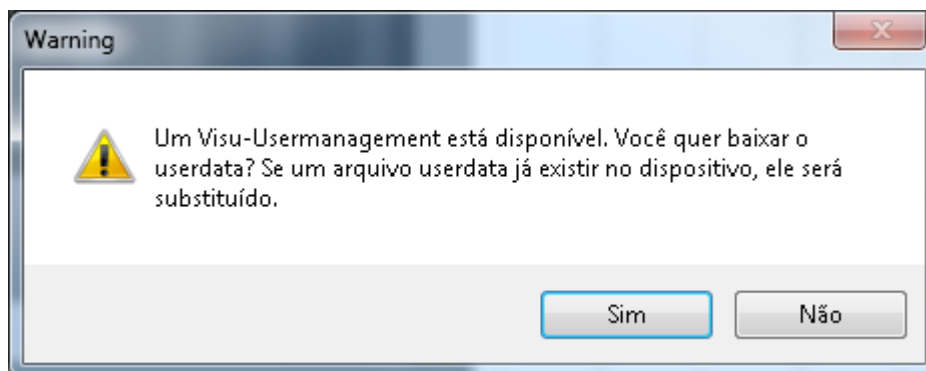


Figure 8-226. Concordar este dialog para baixar

Access rights for elements:

Use group hierarchy: the groups are organized hierarchically in accordance with their position in their configuration in **Groups**. Mark the checkbox then the access rights can be set hierarchically as well.

The Clients and their Editors

Add a Visualization Client

If supported by the device, following client objects can be inserted via *Add object...* in the device tree below the Visualization Manager when it is selected:

:WebVisu

In the following figure the Visualization Manager below *Application* is responsible for visualizations "Visu_1" and "Visu_2". Web visualizations is supported.

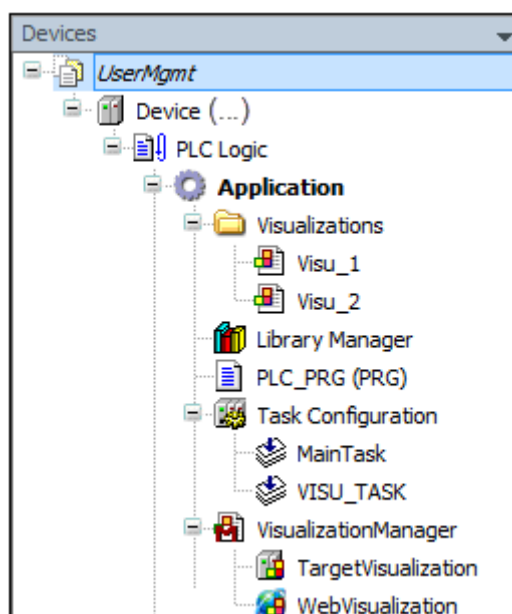


Figure 8-227. Visualization Manager in Device Tree

ATTENTION:

The Diagnosis Visualization mode will be used automatically if there are no client objects available below the Visualization Manager.

Diagnosis Visualization

Visualization client running in the programming system: for diagnosis purposes it might be desired to run the application-assigned visualization(s) only within the programming system without the need of downloading visualization code to the device.

This Diagnosis Visualization will be used automatically if there is no client, *WebVisualization*, in the device tree below the Visualization manager. In this case no visualization code is created and transferred to the device. This however implies several restrictions.

Numeric values visualized in a diagnosis visualization element ("%s" in the 'text' property) will be displayed according to the currently set online display mode (binary, decimal, hexadecimal).

Restrictions on Expressions and Monitoring

The Diagnosis Visualization mode only supports expressions which can be handled by the monitoring mechanism of the programming system.

These are

- normal variable accesses like `MainPrg.myPou.nCounter`
- complex accesses as listed in the following; is required:
 - access on an array of scalar data types with one variable used as index (`a[i]`)
 - access on an array of complex data types (structures, function blocks, arrays) with one variable used as index (`a[i].x`)
 - access on a multidimensional array of all sorts of data types with one or several variable indices (`a[i, 1, j].x`)
 - access on an array with constant index [`a[3]`)
 - accesses like listed above, in which simple operators are used for calculations within the index brackets (`a[i+3]`)
 - nested combinations of the complex expressions listed above (`a[i + 4 * j].aInner[j * 3].x`)
- operators supported in index calculations: `+`, `-`, `*`, `/`, `MOD`
- pointer monitoring like `p^.x`
- not supported are methods or function calls except the following: all standard string functions, all type conversion functions like `INT_TO_DWORD`, all operators like `SEL`, `MIN`, `...`, ...

Restrictions on an input actions

Within the input action *Execute ST-Code* only a list of assignments is supported

Example

```
MainPrg.n := 20 * MainPrg.m;
// not allowed
IF MainPrg.n < MAX_COUNT THEN
    MainPrg.n := MainPrg.n + 1;
END_IF
// use the following statement instead:
MainPrg.n := MIN(MAX_COUNT, MainPrg.n + 1);
```

ATTENTION:

If a list of assignments is used, the value on the left side will not get assigned before the next cycle.
A direct subsequent processing in the successive line is not possible.


Visualization interface

Within the interface of a visualization type 'INTERFACE' must not be used.

Data Server

The Diagnosis Visualization is only designated to inspect the current application..

WebVisu

The WebVisu is a client based on a Java applet which communicates with the web server integrated in the runtime system and displays the visualization in any given visualization systems. It can be used if there is a WebVisu object  inserted below the visualization manager. For this it's necessary to use a PLC witch supports the web server.

The Web Visualization is realized as a Java applet which during startup requests the complete visualization paint information from the web server. After that, only the paint changes get transferred cyclically. At a download of a visualization project all files required for the Web Visualization are transferred to the subdirectory /visu on the target system. This concerns the Java applet, the base HTML-page (HTM file) of the visualization and all images needed in the visualization.

Preconditions for a Web visualization

- Target system with appropriate configuration and web server
- Specification of the start visualization and the name of the HTM file to be downloaded (in the Visualization Manager)
- Web-Browser: IExplorer or Mozilla Firefox; plus Java-VRM as from V1.4

Recommendations Concerning Data Security

In order to minimize the risk of data security breaches we recommend the following technical and organizational measurements for the system running your applications:

- As far as possible avoid to expose PLCs and PLC networks to public networks and internet.
- For protection use additional security layers like for example VPN for remote access, and install firewall mechanisms.
- Restrict access to authorized people.
- If available, please change default passwords at start-up and modify them frequently.
- Check regularly the effect of these measurements.

If you nevertheless want to publish your Web Visualization, it is highly recommended to give it at least a simple password protection to prevent that anybody can access the functionality of your PLC over internet (for an example see the project SimpleWebvisuLogin.project provided with the standard setup of the programming system). Use the latest versions of Gateway Server and Web Server.

Task VISU_TASK

The VISU_TASK, which automatically gets added to the Task Configuration with adding TargetVisu or WebVisu below the Visualization Manager, will be removed when the last of these client objects was deleted.

ATTENTION:

Freewheeling tasks run as fast as possible, so it is not recommended to use the VISU_TASK as a freewheeling task. When a task cycle finishes, it allows other tasks to process during a time equal to 20% of last Cycle Time, so when cycle time is very small the time to process other tasks is very small too. Then the communication performance in every task will decrease and as communication tasks are low priority tasks in general, they need CPU load available to process, not recommended to use the VISU_TASK as a freewheeling task. For more information see also **Task Configuration**.

ATTENTION:

If use Modbus Symbol Server, the VISU_TASK is concurrent, it is not valid only for Modbus Symbol Server but also there is concurrence between communication drivers and WebVisu.

Configuration of the Target System

Possibly the Web Visualization functionality still has to be configured in the target system. The following entries must be available in the configuration file <target system>.cfg:

```
[ComponentManager]
Component.<n>=CmpWebServerHandlerV3
Component.<n>=CmpWebServer
[CmpWebServer]
FileUploadDirectory=C:\Program Files (x86)\Altus\MT8500\MT8500\Common\visu
```

FileUploadDirectory must describe the path of the runtime system, in which the Web Server is available. \visu must be appended to the path. There all files required for the Web-Visualization will be transferred.

Calling a Web Visualization via Internet


Enter the following address in the web browser:

http://<Endereço IP do servidor web>:<porta do servidor web>/<webvisu>.htm

Example: http://localhost:8080/webvisu.htm

<webvisu>.htm is the HTM file defined in the Visualization Manager, via which after the call the start visualization - also defined in the Visualization Manager - will be displayed in the browser window. Thereafter the visualization can be operated in the browser.

Editor for Configuration in the Programming System

The WebVisu object  inserted below the Visualization Manager object contains the settings for the Web Visualization. They can be modified in the editor which opens on a double-click on the object.

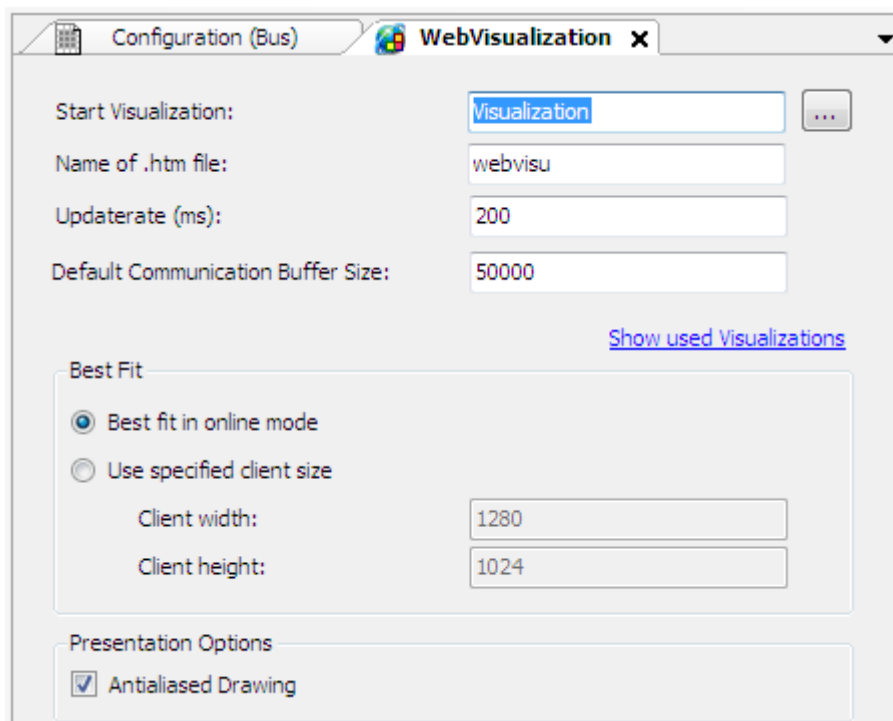


Figure 8-228. Web Visualization Editor

Start Visualization: name of the visualization to be displayed automatically when the Web Visualization gets started. By default Visualization is entered here. To set another visualization you might use the Input Assistant offering the currently available visualizations.

Name of .htm file: enter the name of the basic HTML page of the visualization. This name must also be entered as address in your web browser like `http://localhost:8080/webvisu.htm`.

Updaterate (ms): refresh rate inside the web browser, in milliseconds.

ATTENTION:

If the time set for the Updaterate is smaller than the task interval VISU_TASK will occur the following error: Invalid value for an update rate configured. To adjust the time interval of the task at the same time of the updatarate use the option Adjust the task VISU_TASK to CycleTime... which becomes visible along with the error. This error occurs because it is not convenient to update two or more times the Visualization page while the Visualization is not updated.

Default Communication Buffer Size: defines the maximum available buffer for the data transfer between web client and web browser in Bytes. If this field value is less than 800000, the used value will be 800000 which is defined in the CPU software. This is the reason why is not possible verify any performance change when this field value is changed.

Show used Visualizations: the Visualization Manager can be opened in order to configure the selection of visualizations available for the Web Visualization.

Best Fit:

- *Best fit in online mode*: if this option is activated, the size of the visualization will be adapted to the size of the web-browser window.
- *Use specified client size*: If this option is activated, the size of the visualization will be defined by the following settings:
 - *Client height*: height, in pixel
 - *Client width*: width, in pixel

Presentation Options:

- *Antialiased Drawing*: activate this checkbox if antialiasing should be used when drawing the visualizations within the visualization editor view of the programming system (offline or online).

Visualization in Online Mode

For operating a visualization in online mode via keyboard there are some standard keys supported by each device. Additionally there might be device-specifically supported keys.

Keyboard Usage in Online Mode

Standard Keys

Key	Action
[Tab]	Select next element according to the chronological order of insertion; within a table element each particular cell will be selected; when a frame is selected the selection will be forwarded to the particular subelements
[Shift + Tab]	Select previous element; inverse direction as via [Tab]
[Enter]	Execute input action on the selected element
[Arrow keys]	Select nearest element in the direction given by the arrow key

Table 8-125. Default Hotkeys

For the **Diagnosis Visualization**, the keyboard usage can be explicitly activated and deactivated by command **Activate Keyboard Usage**. This might be useful, because as long as the visualization keyboard operation is activated, any other commands given via key shortcuts will not be executed.

Device-specific Keyboard Usage

Besides the standard shortcuts described above it depends on the device which further keys might be used for the visualizations running on that device. For details see the **Default Hotkeys**, valid for all visualizations below an application, **Hotkeys Configuration**.

There is the possibility to handle key events within the application code: **Notice Key Events**.

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.

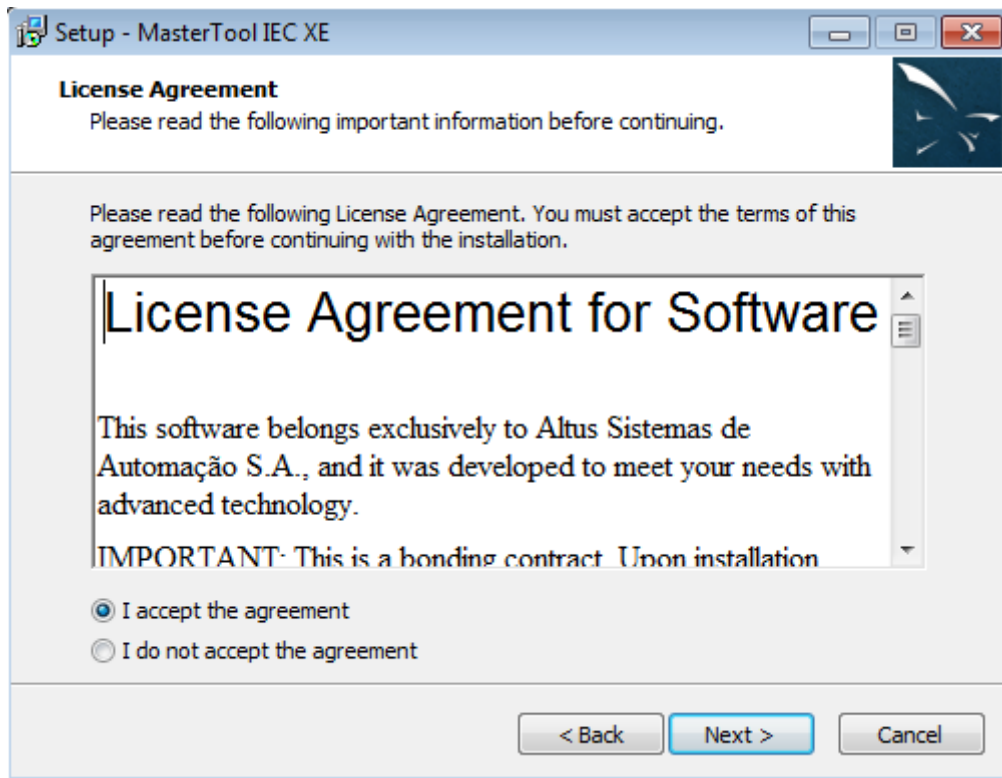


Figure 9-3. License Screen

It will appear a screen with information regarding the MasterTool IEC XE new version that have just been installed.

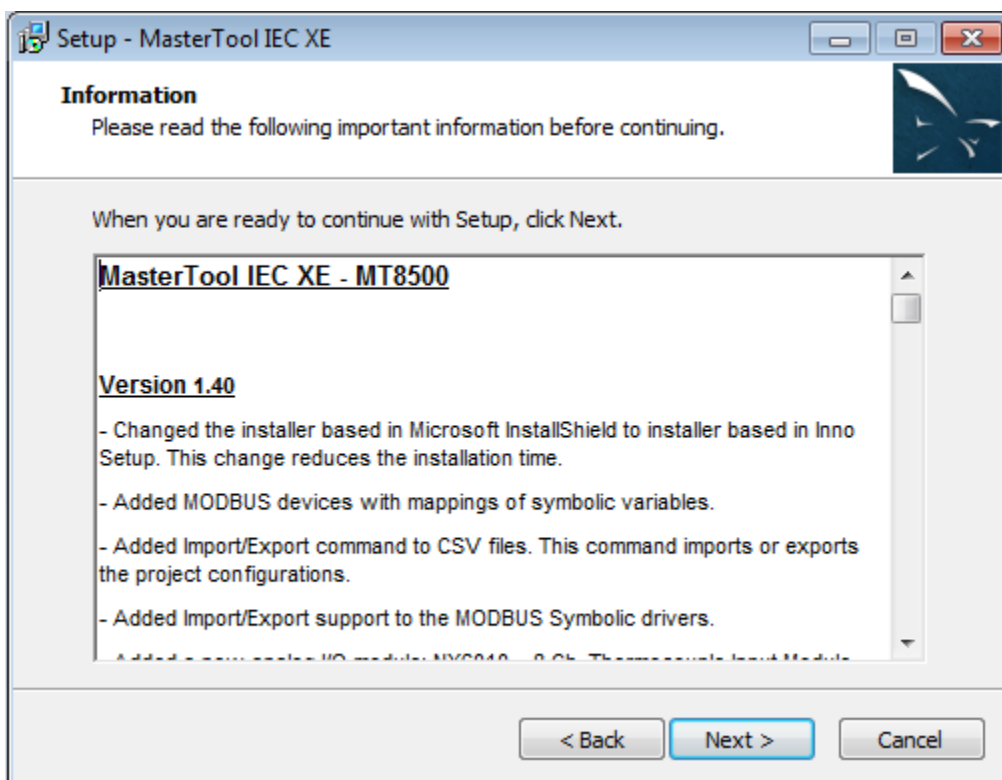


Figure 9-4. Version Information Screen

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.

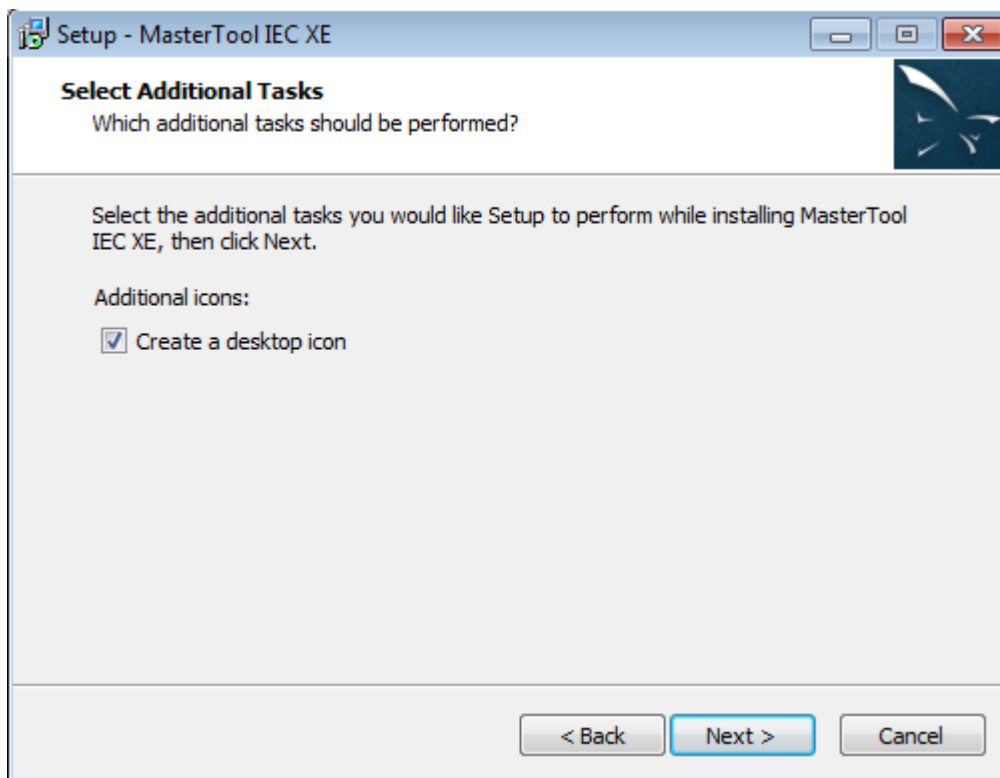


Figure 9-5. Icon on the Desktop Selection Screen

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.

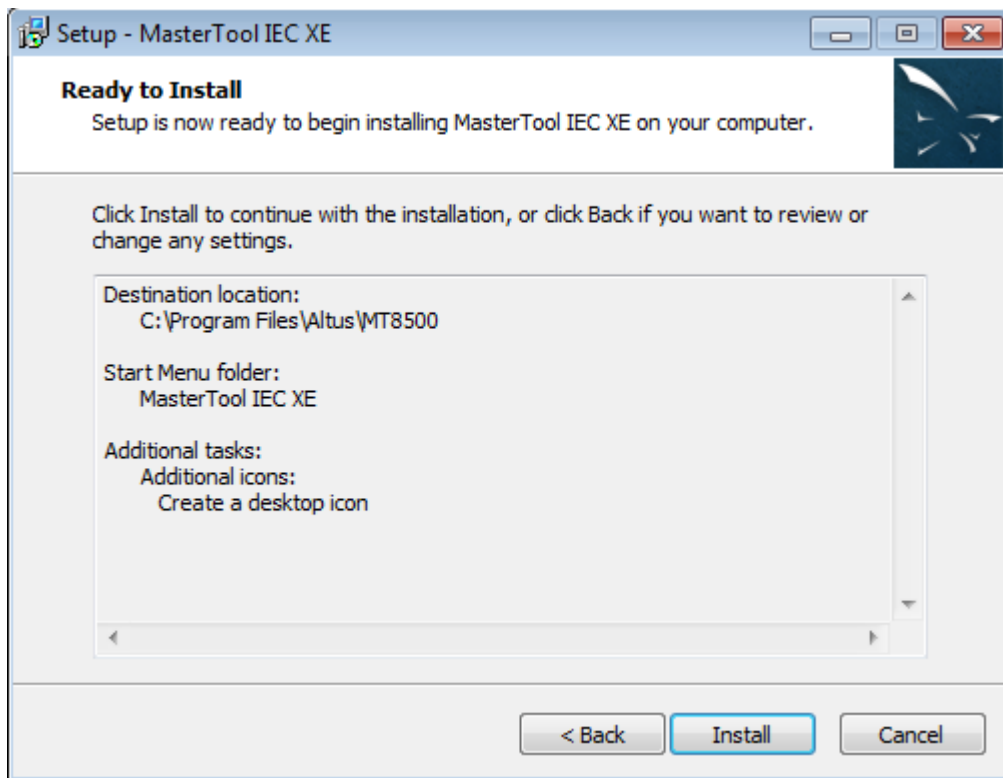


Figure 9-6. Components Revision Screen

At first the software prerequisites, such as the .NET Framework have to be installed.

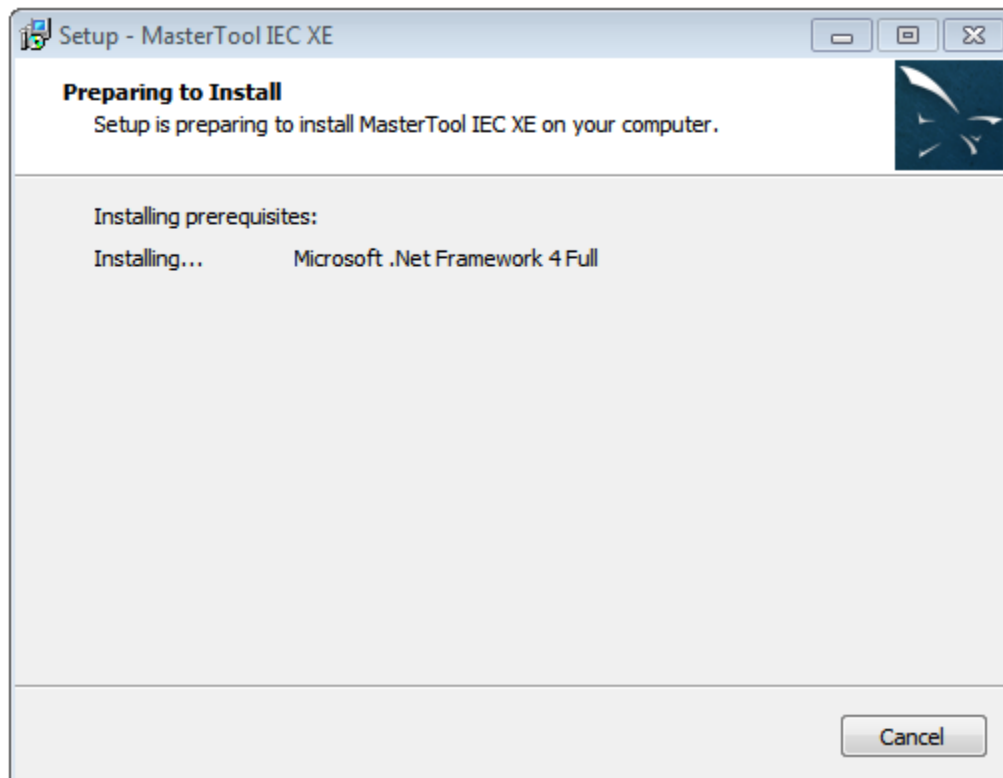


Figure 9-7. Prior Conditions Screen to Install MasterTool IEC XE

During the installation process, a screen requesting the extraction of the .NET Framework files will appear.

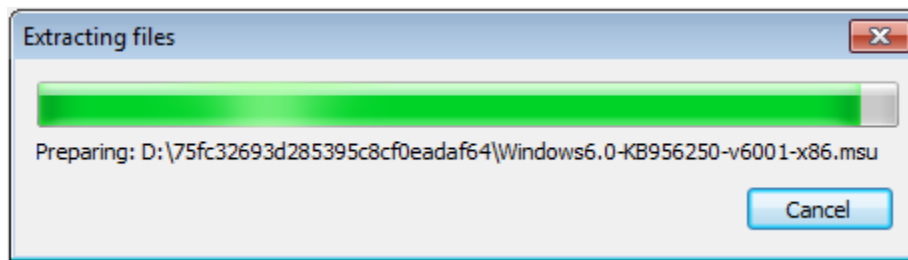


Figure 9-8. Extracting Files for the .NET Framework Installation

The Microsoft .NET Framework installation will be started.

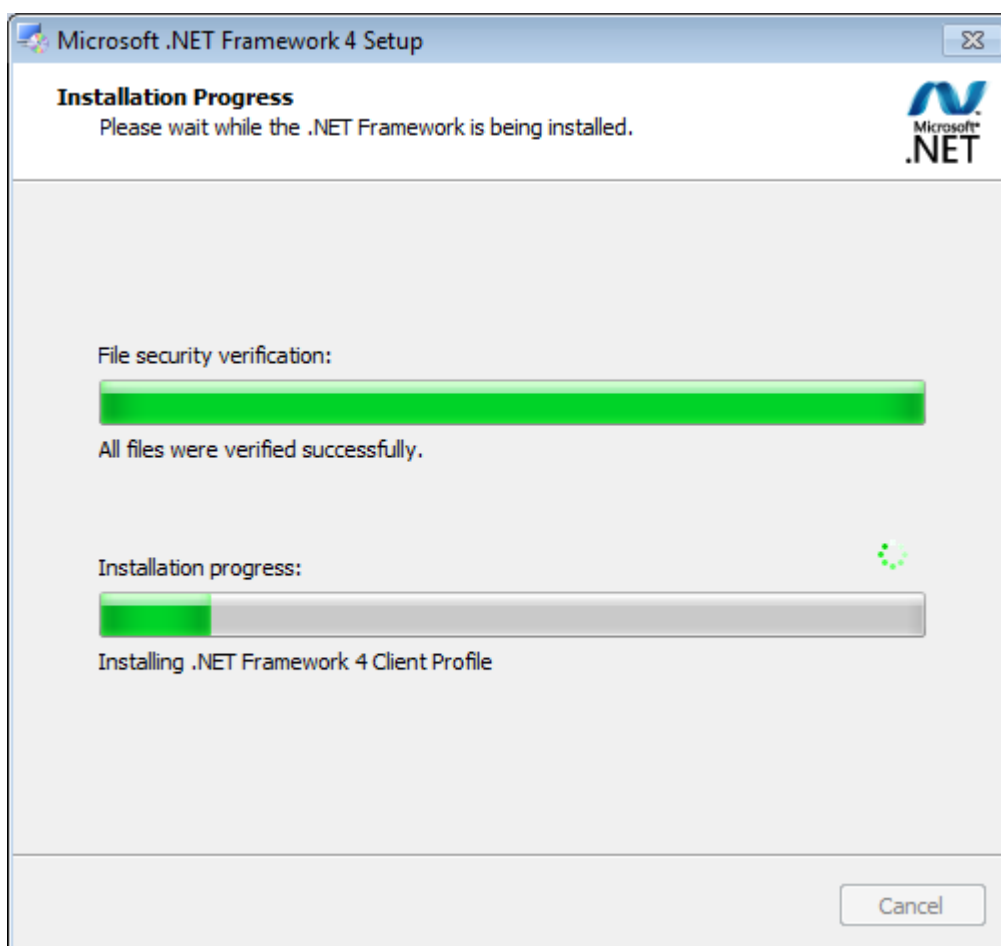


Figure 9-9. .NET Framework Installation Screen

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, depending on its configuration.

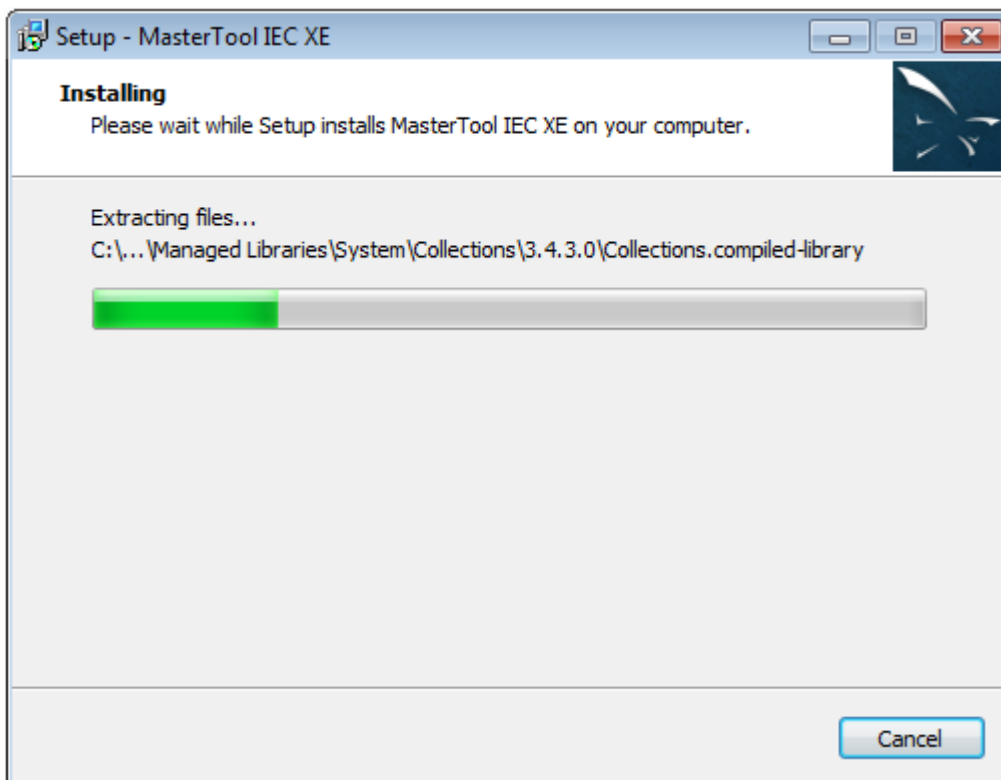


Figure 9-10. MasterTool IEC XE Installation Screen

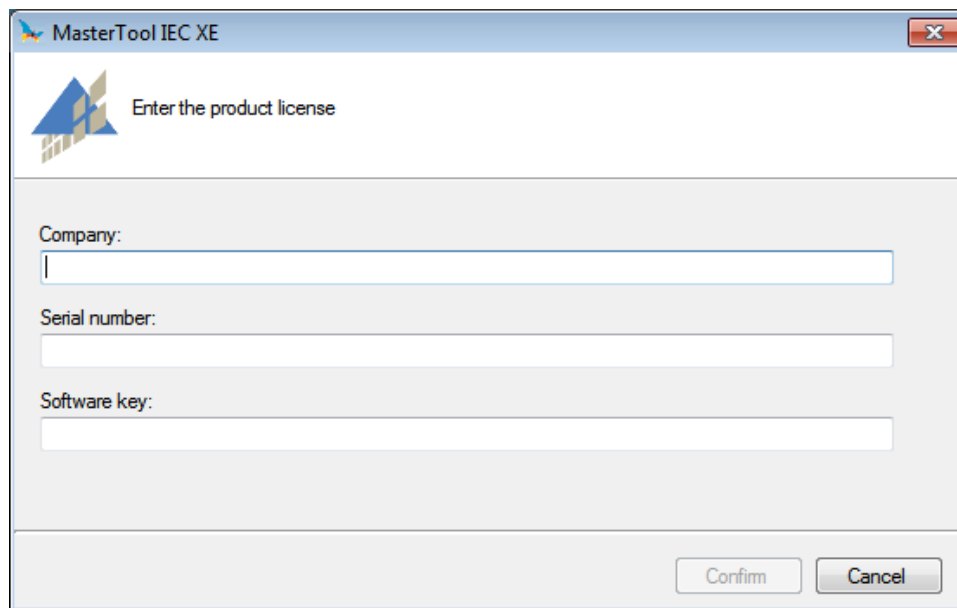
After installation is finished the next screen will be shown. On this window it is strongly recommendable to choose the restart option. After that, click *Finish* to conclude the installation procedure.



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.



The image shows a Windows-style dialog box titled "MasterTool IEC XE". The dialog has a blue header bar with a close button (X) in the top right corner. Below the header, there is a logo consisting of several blue and grey geometric shapes, followed by the text "Enter the product license". The main area of the dialog is light grey and contains three text input fields. The first field is labeled "Company:" and is empty. The second field is labeled "Serial number:" and is empty. The third field is labeled "Software key:" and is empty. At the bottom right of the dialog, there are two buttons: "Confirm" and "Cancel".

Figure 9-12. Registry Information Screen

NOTE: 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**).

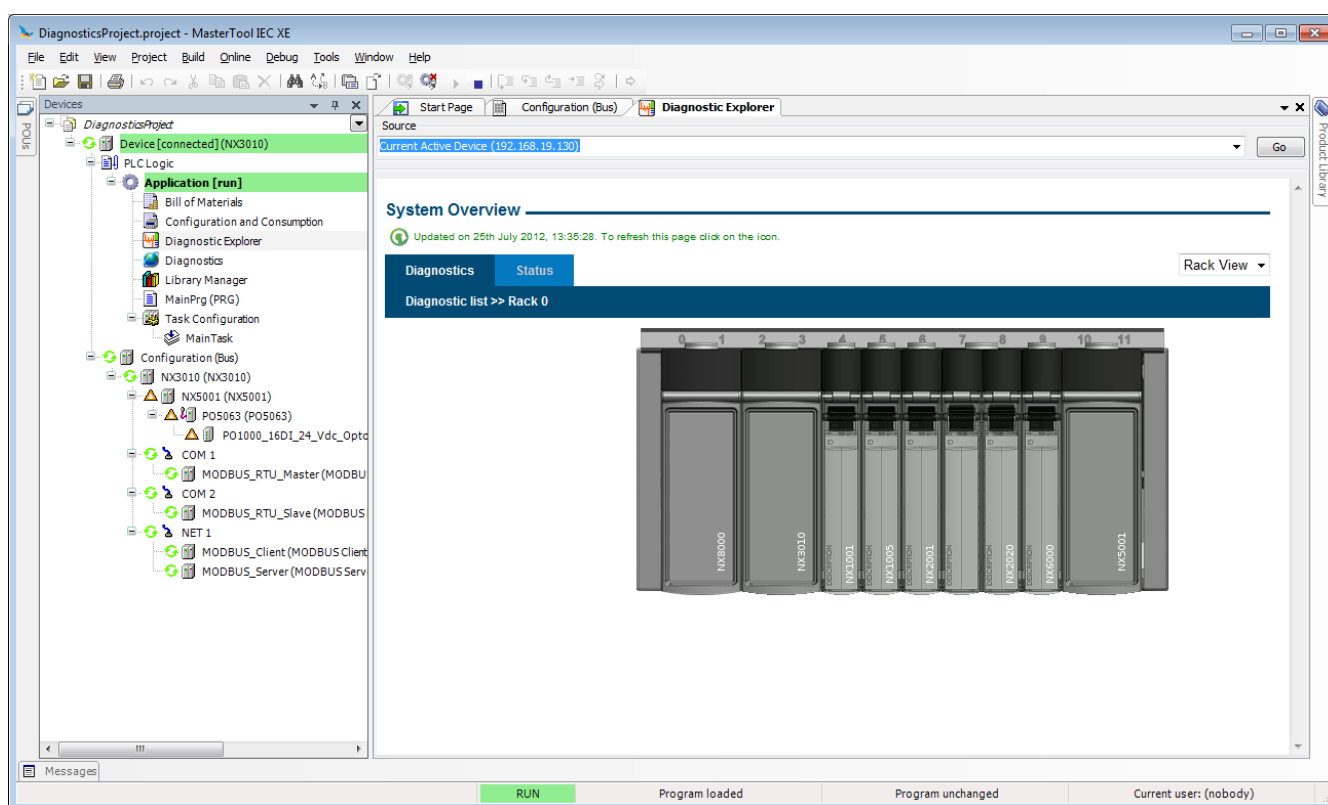


Figure 10-1. Diagnostics Explorer

- 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.

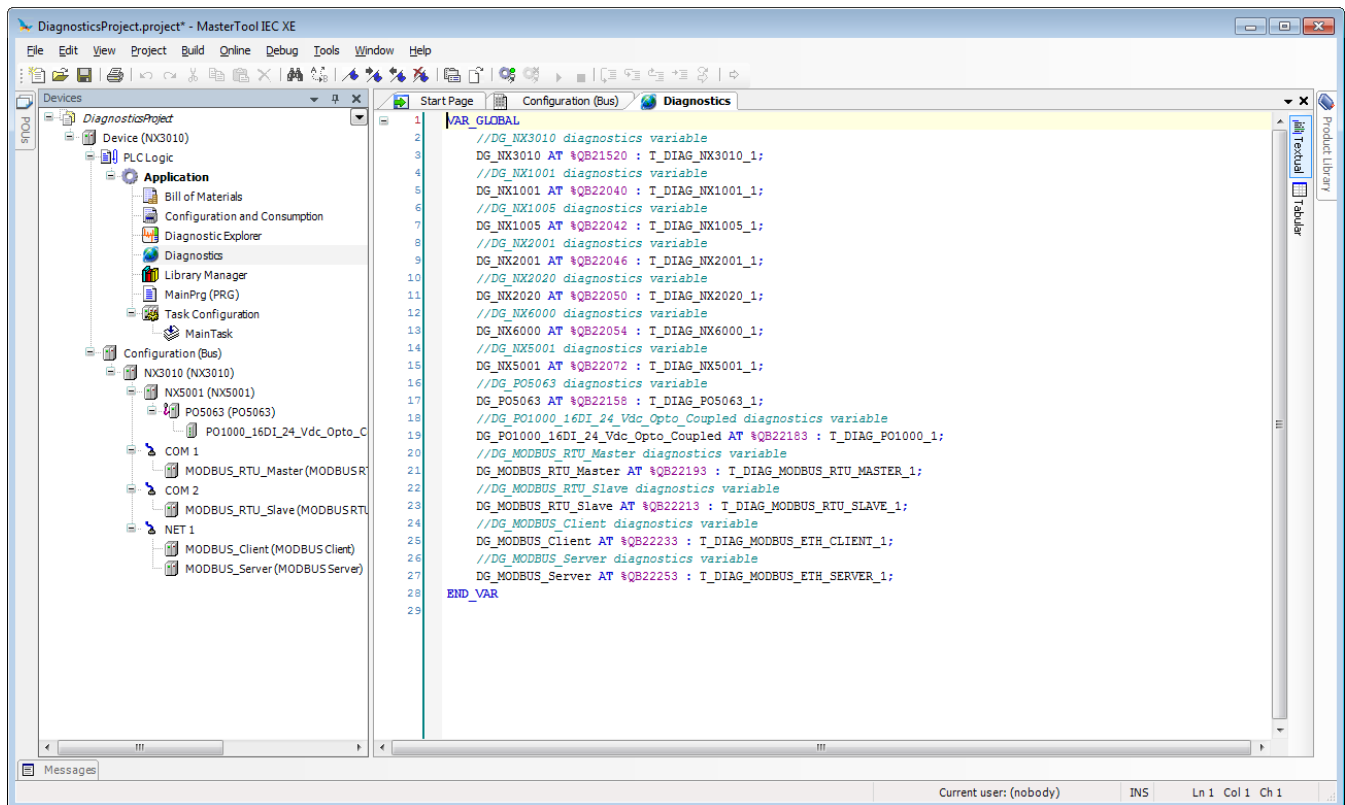


Figure 10-2. Diagnostics Variables

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.

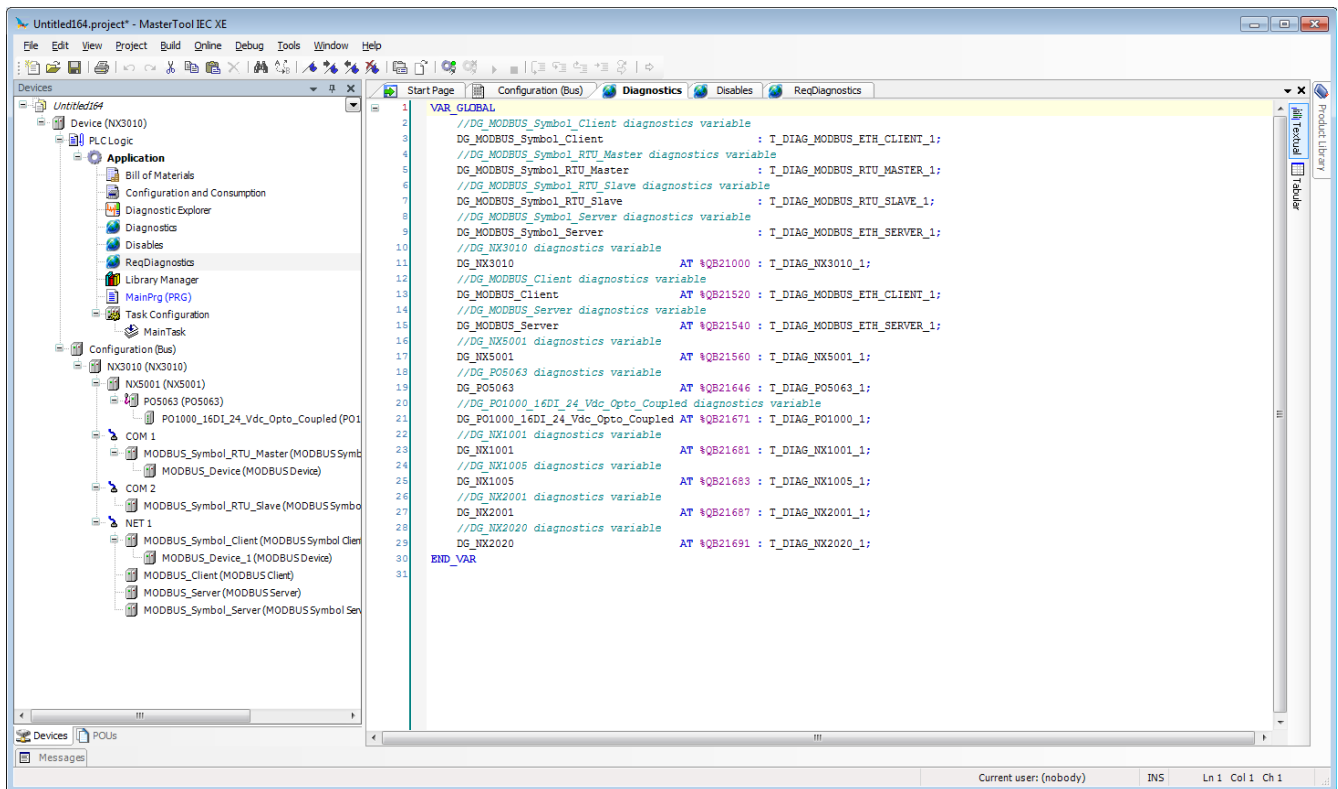


Figure 10-3. Diagnostics Variables Using Communication Drivers Through Symbolic Mapping

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.

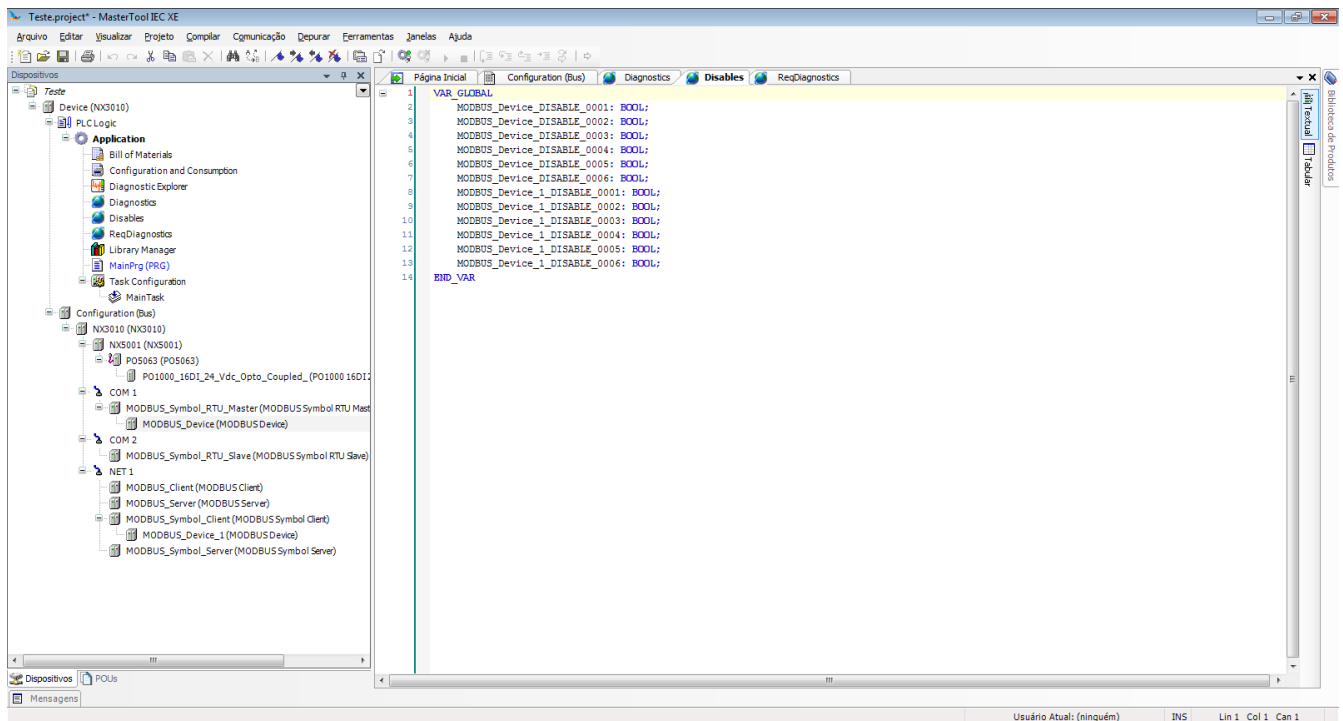


Figure 10-4. Disables Object

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.

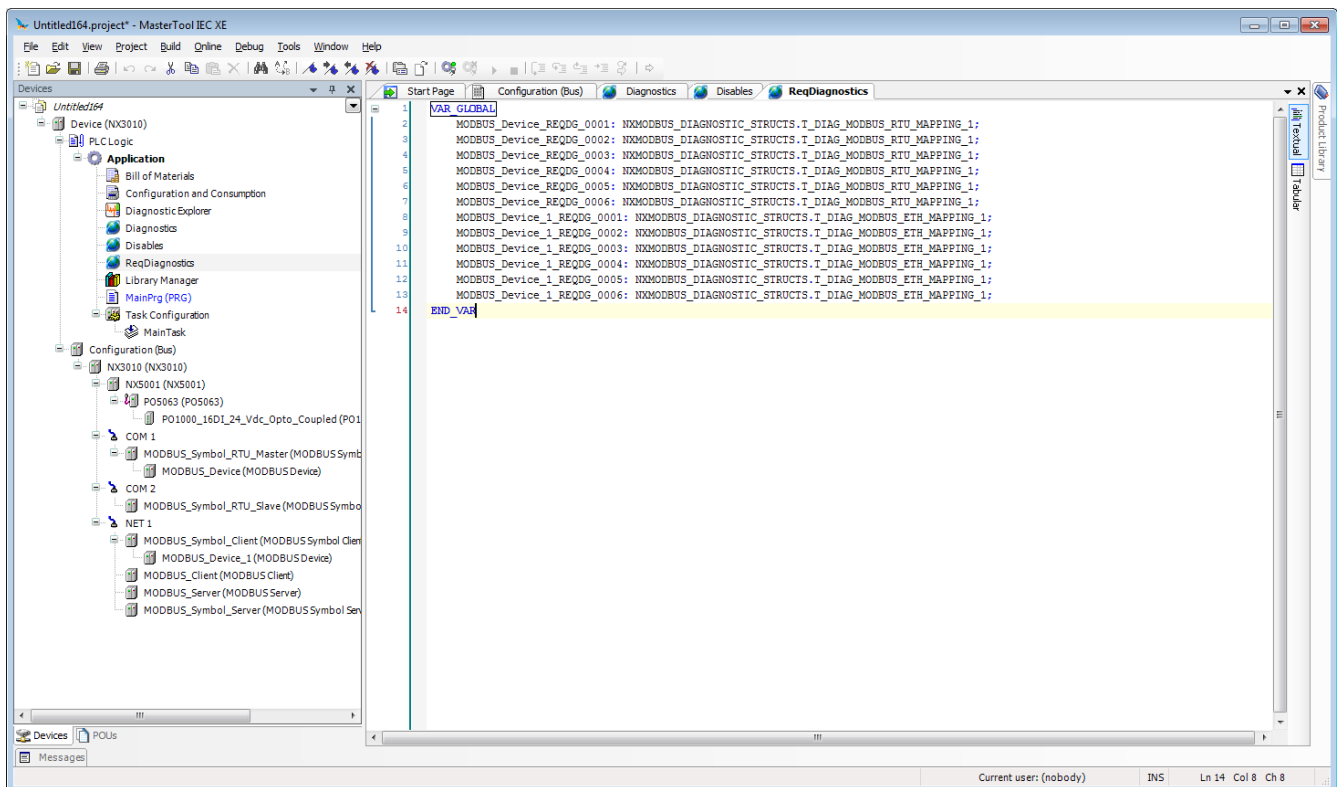


Figure 10-5. ReqDiagnostics Object

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 latter 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.