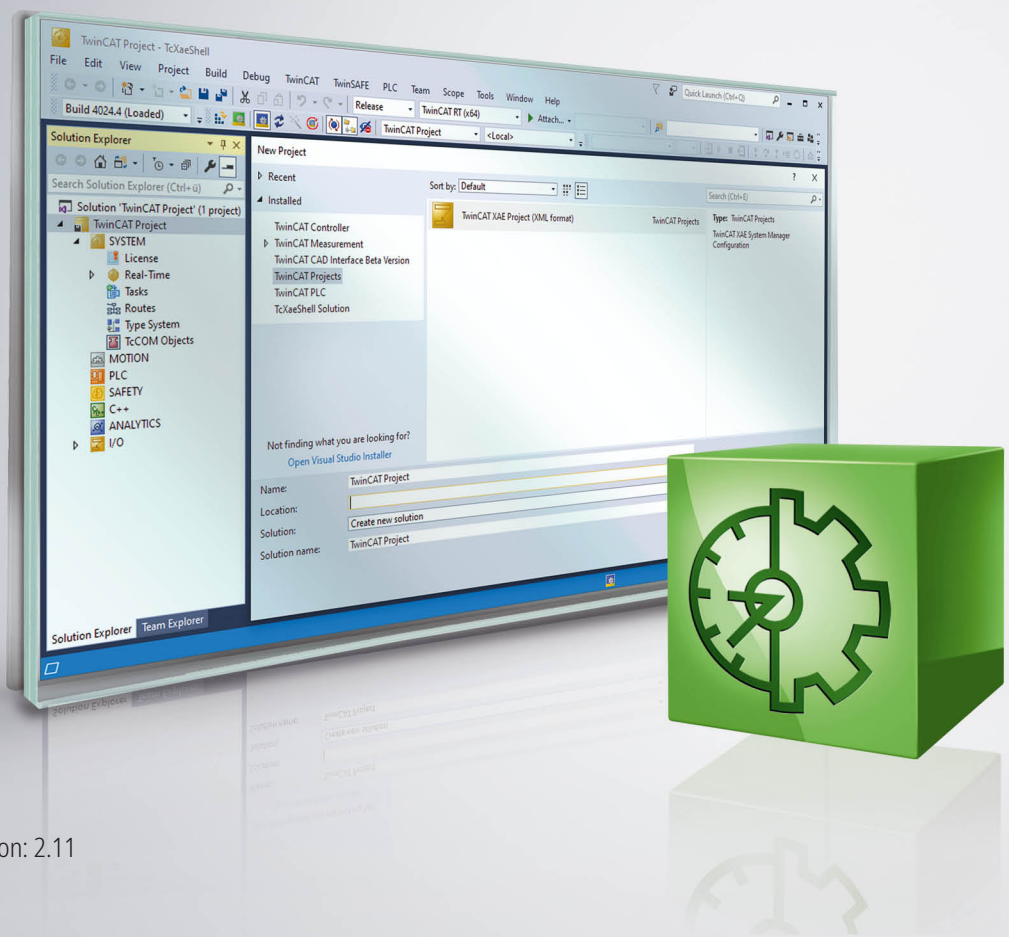
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | User Interface



2021-08-30 | Version: 2.11

Table of contents

1 Foreword 13
1.1 Notes on the documentation 13
1.2 Safety instructions 14

2 User Interface components 15
2.1 Standard menus 17
2.2 Standard toolbars 17
2.3 Standard commands 18
2.4 Default views and windows 26
2.5 Information bar 27
2.6 Status symbols 28
2.7 System menu 28
2.7.1 About TwinCAT... 29
2.7.2 Realtime 29
2.7.3 Router 30
2.7.4 System 31
2.7.5 Tools 31
2.8 User interface in online mode 32

3 Configuring user interface 33
3.1 Setting options 33
3.1.1 Customizing text editor settings 33
3.1.2 Select language 34
3.2 Customizing the user interface 34
3.2.1 Customizing Menus 34
3.2.2 Customizing Toolbars 35
3.2.3 Customizing Keyboard Shortcuts 36
3.2.4 Arranging the menu bar and the toolbar 36
3.2.5 Arranging views and windows 37
3.2.6 Toggle between views and windows 38
3.2.7 Show/hide views 38
3.2.8 Resizing Windows 38

4 Use TwinCAT Documentation 40
4.1 Calling the TwinCAT 3 documentations 40
4.2 Updating the TwinCAT 3 documentation 44
4.2.1 Update in Visual Studio 44
4.2.2 Update in Visual Studio 2010 46

5 Reference User Interface 50
5.1 File 50
5.1.1 Archiving options 50
5.1.2 Command Project... (Create new TwinCAT project) 56
5.1.3 Command Project/Solution (Open Project/Solution) 58
5.1.4 Command Open Project from Target 59
5.1.5 Command New Project... (Add new TwinCAT project) 59
5.1.6 Command Existing Item... (Add existing TwinCAT project) 59

5.1.7 Command Recent Projects and Solutions ... 59
5.1.8 Command Save All ... 59
5.1.9 Command Save ... 60
5.1.10 Command Save <Solution name> as ... 60
5.1.11 Command Save <TwinCAT project name> as ... 60
5.1.12 Command Save <PLC project name> as ... 60
5.1.13 Command Send by E-Mail... ... 61
5.1.14 Command Close Solution ... 61
5.1.15 Command Close ... 61
5.1.16 Command Exit ... 61
5.1.17 Command Page Setup... ... 62
5.1.18 Command Print ... 62
5.2 Edit ... 62
5.2.1 Standard Commands ... 62
5.2.2 Command Delete ... 63
5.2.3 Command Select All ... 63
5.2.4 Command Input Assistant ... 63
5.2.5 Command Auto Declare ... 65
5.2.6 Command Add to Watch ... 68
5.2.7 Command Browse Call Tree ... 68
5.2.8 Command Go To ... 68
5.2.9 Command Go To Definition ... 68
5.2.10 Command Go to Instance ... 69
5.2.11 Command Find all references ... 69
5.2.12 Command Navigate To ... 69
5.2.13 Command Make Uppercase ... 69
5.2.14 Command Make Lowercase ... 70
5.2.15 Command View white spaces ... 70
5.2.16 Command Comment Selection ... 70
5.2.17 Command Uncomment Selection ... 70
5.2.18 Command Quick Find ... 70
5.2.19 Command Quick Replace ... 72
5.2.20 Command Switch write mode ... 74
5.2.21 Command Rename ... 74
5.2.22 Command Edit object (offline) ... 74
5.2.23 Command Rename '<variable>' ... 74
5.2.24 Command Add '<variable>' ... 75
5.2.25 Command Remove '<variable>' ... 77
5.2.26 Command Reorder variables ... 78
5.3 View ... 78
5.3.1 Command Open object ... 78
5.3.2 Command Textual view ... 79
5.3.3 Command Tabular view ... 79
5.3.4 Command Full screen ... 80
5.3.5 Command Toolbars ... 80
5.3.6 Command Solution Explorer ... 80

5.3.7 Command Properties Window ... 81
5.3.8 Command Toolbox ... 82
5.3.9 Command Error List ... 83
5.3.10 Command Output ... 84
5.4 Project ... 85
5.4.1 Command Add New Item (project) ... 85
5.4.2 Command Add Existing Item (Project) ... 86
5.4.3 Command Properties (object) ... 90
5.4.4 Command Properties (PLC project) ... 94
5.4.5 PLC project settings ... 110
5.5 Build ... 114
5.5.1 Command Build Solution ... 114
5.5.2 Command Rebuild Solution ... 114
5.5.3 Command Clean Solution ... 114
5.5.4 Command Check all objects ... 114
5.5.5 Command Build TwinCAT project ... 115
5.5.6 Command Rebuild a TwinCAT project ... 115
5.5.7 Command Clean TwinCAT project ... 115
5.5.8 Command Build PLC project ... 115
5.5.9 Command Rebuild a PLC project ... 116
5.5.10 Command Clean PLC project ... 116
5.6 Debug ... 116
5.6.1 Command New Breakpoint ... 116
5.6.2 Command Edit Breakpoint ... 120
5.6.3 Command Enable Breakpoint ... 120
5.6.4 Command Disable Breakpoint ... 120
5.6.5 Command Toggle Breakpoint ... 120
5.6.6 Command Step over ... 121
5.6.7 Command Step into ... 121
5.6.8 Command Step out ... 121
5.6.9 Command Show Next Statement ... 122
5.6.10 Command Set next statement ... 122
5.6.11 Command Run To Cursor ... 122
5.7 TwinCAT ... 123
5.7.1 Command Activate configuration ... 123
5.7.2 Command Restart TwinCAT System ... 123
5.7.3 Command Restart TwinCAT (Config mode) ... 123
5.7.4 Command Reload Devices ... 123
5.7.5 Command Scan ... 123
5.7.6 Command Toggle Free Run State ... 124
5.7.7 Command Show Online Data ... 124
5.7.8 Command Choose Target System ... 124
5.7.9 Command Show Sub Items ... 124
5.7.10 Command Software Protection ... 125
5.7.11 Command Hide Disabled Items ... 125
5.8 PLC ... 125

5.8.1 Window ... 125
5.8.2 Core dump ... 135
5.8.3 Command Download ... 137
5.8.4 Command Online Change ... 138
5.8.5 Command Login ... 139
5.8.6 Command Start ... 140
5.8.7 Command Stop ... 140
5.8.8 Command Logout ... 140
5.8.9 Command Reset cold ... 140
5.8.10 Command Reset origin ... 141
5.8.11 Command Single cycle ... 141
5.8.12 Command Flow Control ... 141
5.8.13 Command Force values ... 142
5.8.14 Command Unforce values ... 143
5.8.15 Command Write values ... 144
5.8.16 Command Display Mode - Binary, Decimal, Hexadecimal ... 145
5.8.17 Command Create Localization Template ... 145
5.8.18 Command Manage Localization ... 145
5.8.19 Command Toggle Localization ... 146
5.8.20 Command Active PLC project ... 146
5.8.21 Command Active PLC instance ... 146
5.9 Tools ... 147
5.9.1 Command Options ... 147
5.9.2 Command Customize ... 172
5.10 Window ... 175
5.10.1 Command Float ... 175
5.10.2 Command Dock ... 175
5.10.3 Command Hide ... 175
5.10.4 Command Auto Hide All ... 176
5.10.5 Command Auto Hide ... 176
5.10.6 Command Pin tab ... 176
5.10.7 Command New Horizontal Tab Group ... 177
5.10.8 Command New Vertical Tab Group ... 177
5.10.9 Command Reset Window-Layout ... 177
5.10.10 Command Close All Documents ... 177
5.10.11 Command Window ... 178
5.10.12 Window submenu commands ... 178
5.11 SFC ... 178
5.11.1 Command Init step ... 178
5.11.2 Command Insert step transition ... 178
5.11.3 Command Insert step-transition after ... 179
5.11.4 Command Parallel ... 179
5.11.5 Command Alternative ... 180
5.11.6 Command Insert Branch ... 180
5.11.7 Command Insert branch right ... 180
5.11.8 Command Insert action association ... 181

5.11.9 Command Insert action association after ... 182
5.11.10 Command Insert jump ... 182
5.11.11 Command Insert jump after ... 183
5.11.12 Command Insert macro ... 183
5.11.13 Command Insert macro after ... 183
5.11.14 Command Show macro ... 184
5.11.15 Command Exit macro ... 184
5.11.16 Command Insert after ... 184
5.11.17 Command Add entry action ... 185
5.11.18 Command Add exit action ... 186
5.11.19 Command Change duplication - Set ... 186
5.11.20 Command Change duplication - Remove ... 186
5.11.21 Command Insert step ... 187
5.11.22 Command Insert step after ... 187
5.11.23 Command Insert transition ... 187
5.11.24 Command Insert transition after ... 188
5.12 CFC ... 188
5.12.1 Command Edit Worksheet ... 188
5.12.2 Command Negate ... 189
5.12.3 Command EN/ENO ... 189
5.12.4 Command None ... 190
5.12.5 Command R-Reset ... 190
5.12.6 Command S-SET ... 190
5.12.7 Command REF = (reference assignment) ... 191
5.12.8 Command Move to Beginning ... 191
5.12.9 Command Move to End ... 191
5.12.10 Command Forward by one ... 192
5.12.11 Command Back by one ... 192
5.12.12 Command Order by Data Flow ... 192
5.12.13 Command Order By Topology ... 193
5.12.14 Command Set Execution Order ... 193
5.12.15 Command Connect Selected Pins ... 193
5.12.16 Command Reset Pins ... 194
5.12.17 Command Remove Unused Pins ... 194
5.12.18 Command Add Input Pin ... 194
5.12.19 Command Add Output Pin ... 195
5.12.20 Command Route All Connections ... 195
5.12.21 Command Show Next Collision ... 195
5.12.22 Command Remove Control Point ... 196
5.12.23 Command Create Control Point ... 196
5.12.24 Command Unlock Connection ... 196
5.12.25 Command Create group ... 197
5.12.26 Command Ungroup ... 197
5.12.27 Command Edit page size ... 197
5.12.28 Command Edit Parameters ... 198
5.12.29 Command Save prepared parameters in the project ... 199

5.12.30 Command Connection Mark ... 199
5.12.31 Command Select Connected Pins ... 200
5.13 FBD/LD/IL ... 200
5.13.1 Command Insert Contact (right) ... 200
5.13.2 Command Insert Network ... 200
5.13.3 Command Insert Network (below) ... 200
5.13.4 Command Toggle comment state ... 201
5.13.5 Command Insert Assignment ... 201
5.13.6 Command Insert Box ... 201
5.13.7 Command Insert Box with EN/ENO ... 202
5.13.8 Command Insert Empty Box ... 202
5.13.9 Command Insert Box with EN/ENO ... 202
5.13.10 Command Insert jump ... 203
5.13.11 Command Insert label ... 203
5.13.12 Command Insert Return ... 203
5.13.13 Command Insert Input ... 203
5.13.14 Command Insert box in parallel (below) ... 204
5.13.15 Command Insert Coil ... 204
5.13.16 Command Insert Set coil ... 204
5.13.17 Command Insert Reset coil ... 205
5.13.18 Command Insert Contact ... 205
5.13.19 Command Insert Contact Parallel (below) ... 205
5.13.20 Command Insert Contact Parallel (above) ... 205
5.13.21 Command Insert Negated Contact ... 206
5.13.22 Command Insert Negated Contact Parallel (below) ... 206
5.13.23 Command Paste Contacts: Paste below ... 206
5.13.24 Command Paste Contacts: Paste above ... 207
5.13.25 Command Paste Contacts: Paste right (after) ... 207
5.13.26 Command Insert IL line below ... 207
5.13.27 Command Delete IL line ... 207
5.13.28 Command Negation ... 208
5.13.29 Command Edge detection ... 208
5.13.30 Command Set/Reset ... 208
5.13.31 Command Set output connection ... 209
5.13.32 Command Insert Branch ... 209
5.13.33 Command Insert Branch above ... 209
5.13.34 Command Insert Branch below ... 209
5.13.35 Command Set Branch Start Point ... 210
5.13.36 Command Set Branch End Point ... 210
5.13.37 Command Update parameters ... 210
5.13.38 Command Remove unused FB call parameters ... 211
5.13.39 Command Repair POU ... 211
5.13.40 Command View as function block diagram ... 211
5.13.41 Command View as ladder logic ... 212
5.13.42 Command View as instruction list ... 212
5.13.43 Command Go To ... 212

5.14 Textlist ... 213
5.14.1 Command Add Language ... 213
5.14.2 Command Remove Language ... 213
5.14.3 Command Insert Text ... 213
5.14.4 Command Import/Export Text Lists ... 214
5.14.5 Command Remove Unused Text List Records ... 215
5.14.6 Command Check Visualization Text Ids ... 215
5.14.7 Command Update Visualization Text Ids ... 216
5.14.8 Command Export All ... 216
5.14.9 Command Export All Unicode ... 216
5.14.10 Command Add text list support ... 217
5.14.11 Command Remove text list support ... 217
5.15 Recipes ... 217
5.15.1 Command Add a new recipe ... 217
5.15.2 Command Remove recipe ... 218
5.15.3 Command Load Recipe ... 218
5.15.4 Command Save recipe ... 219
5.15.5 Command Read Recipe ... 219
5.15.6 Command Write Recipe ... 219
5.15.7 Command Load and Write Recipe ... 220
5.15.8 Command Read and Save Recipe ... 220
5.15.9 Command Insert variable ... 220
5.15.10 Command Remove variables ... 221
5.15.11 Command Update structured variables ... 221
5.15.12 Command Download recipes from the device ... 222
5.16 Library ... 223
5.17 Visualization ... 223
5.17.1 Command Interface Editor ... 223
5.17.2 Command Hotkey Configuration ... 223
5.17.3 Command Element List ... 224
5.17.4 Command Align Left ... 224
5.17.5 Command Align Top ... 224
5.17.6 Command Align Right ... 224
5.17.7 Command Align Bottom ... 224
5.17.8 Command Align Vertical Center ... 224
5.17.9 Command Align Horizontal Center ... 225
5.17.10 Command Make horizontal spacing equal ... 225
5.17.11 Command Increase horizontal spacing ... 225
5.17.12 Command Decrease horizontal spacing ... 225
5.17.13 Command Remove horizontal spacing ... 225
5.17.14 Command Make vertical spacing equal ... 226
5.17.15 Command Increase vertical spacing ... 226
5.17.16 Command Decrease vertical spacing ... 226
5.17.17 Command Remove vertical spacing ... 226
5.17.18 Command Make same width ... 227
5.17.19 Command Make same height ... 227

5.17.20 Command Make same size ... 227
5.17.21 Command Size to Grid ... 227
5.17.22 Command Bring One to Front ... 228
5.17.23 Command Bring to front ... 228
5.17.24 Command Send One to Back ... 228
5.17.25 Command Send to Back ... 228
5.17.26 Command Group ... 228
5.17.27 Command Ungroup ... 229
5.17.28 Command Background ... 229
5.17.29 Command Select All ... 229
5.17.30 Command Deselect All ... 230
5.17.31 Command Multiply visu element ... 230
5.17.32 Command Activate keyboard usage ... 231
5.18 Miscellaneous ... 231
5.18.1 Command Implement interfaces ... 231
5.19 Context menu TwinCAT project ... 232
5.19.1 Command Save <TwinCAT project name> as Archive... ... 232
5.19.2 Command Send by E-Mail... ... 233
5.19.3 Command Backup <TwinCAT project names> automatically to the target system ... 233
5.19.4 Command Compare <TwinCAT project name> with the target system ... 233
5.19.5 Command Update project with target system... ... 233
5.19.6 Command Load project with TwinCAT 2.xx Version... ... 233
5.19.7 Command Show Hidden Configurations ... 234
5.19.8 Command Remove From Solution ... 234
5.19.9 Command Rename ... 234
5.19.10 Command Build TwinCAT project ... 234
5.19.11 Command Rebuild a TwinCAT project ... 234
5.19.12 Command Clean TwinCAT project ... 235
5.19.13 Command Unload Project ... 235
5.19.14 Import via AML DataExchange... ... 235
5.19.15 Export AutomationML... ... 235
5.20 PLC project context menu ... 236
5.20.1 Command Activate boot project ... 236
5.20.2 Command Autostart boot project ... 236
5.20.3 Command Change ADS port ... 236
5.20.4 Command Install Project Library ... 236
5.20.5 Command Install Project Library (unknown versions) ... 237
5.20.6 Command Update project library folder ... 238
5.20.7 Command Change project ... 238
5.20.8 Command Add New Item (instance) ... 238
5.20.9 Command Save <PLC project name> as archive ... 239
5.20.10 Command Send by E-Mail... ... 240
5.20.11 Command Update project with target system... ... 240
5.20.12 Command Independent project file ... 240
5.20.13 Command Deactivate ... 240
5.21 Context menu PLC project object (<PLC project name> Project) ... 240

5.21.1 Command Login ... 240
5.21.2 Command Build PLC project ... 242
5.21.3 Command Rebuild a PLC project ... 242
5.21.4 Command Check all objects ... 242
5.21.5 Command Clean PLC project ... 243
5.21.6 Add ... 243
5.21.7 Command Export to ZIP ... 244
5.21.8 Command Import from ZIP ... 245
5.21.9 Command Export PLCopenXML ... 245
5.21.10 Command Import PLCopenXML ... 245
5.21.11 Command Save as library ... 245
5.21.12 Command Save as library and install ... 247
5.21.13 Command Properties (PLC project) ... 248

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

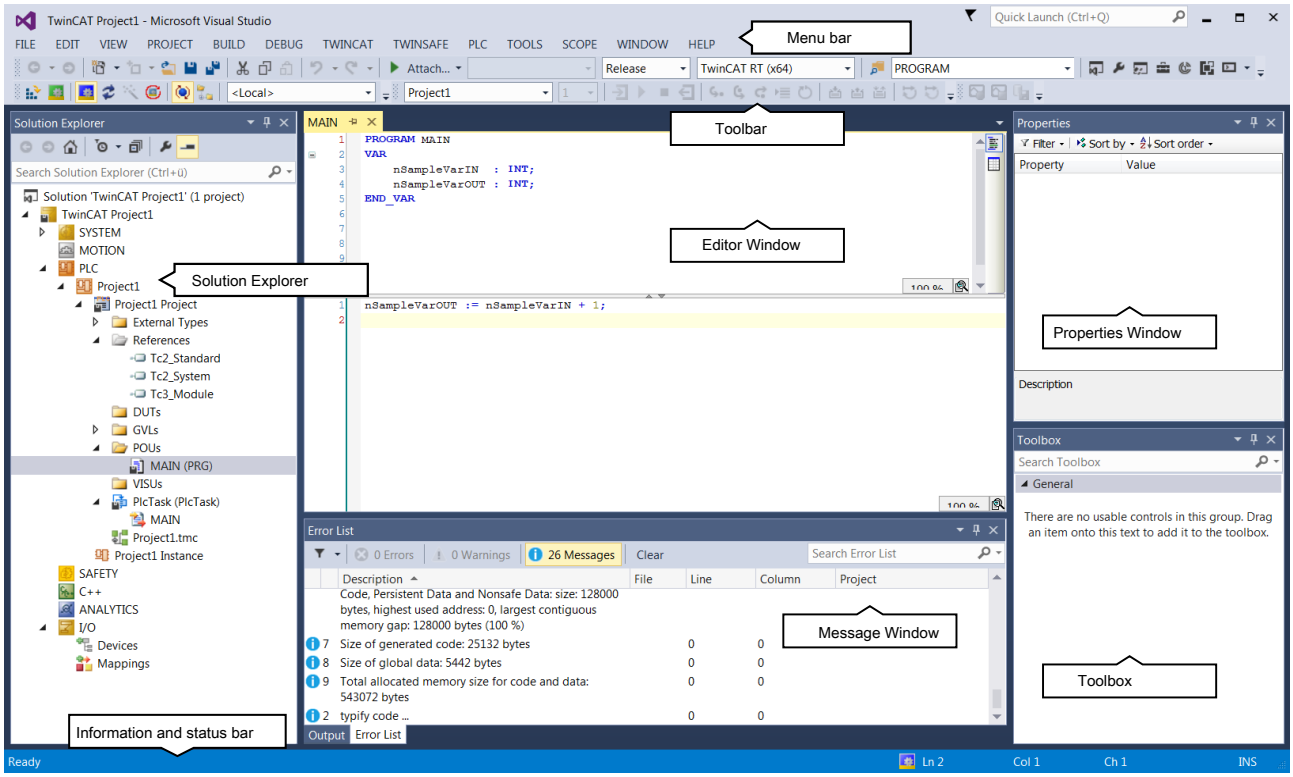
Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

2 User Interface components

TwinCAT 3 Engineering consists of various components. The appearance of the user interface is determined by the arrangement and configuration of the individual components.



Standard components

Menu bar	Shows the menus according to the settings in the **Customize** dialog.
Toolbar	Shows the commands as buttons identified with symbols according to the settings in the **Customize** dialog.
Toolbox	Shows the “tools” that are available for the currently active editor (e.g., graphical programming elements)
Solution Explorer	Shows the TwinCAT 3 project with the associated project elements in a structured form.
Properties Window	Shows the properties of the element that is currently selected in the Solution Explorer.
Editor Window	This is used for defining and editing objects. For language editors (e.g., ST editor, CFC editor), the editor window usually shows the language editor in the lower part (implementation part) and the declaration editor in the upper part. For other editors, the editor window may also contain dialogs (e.g., task editor, device editor).

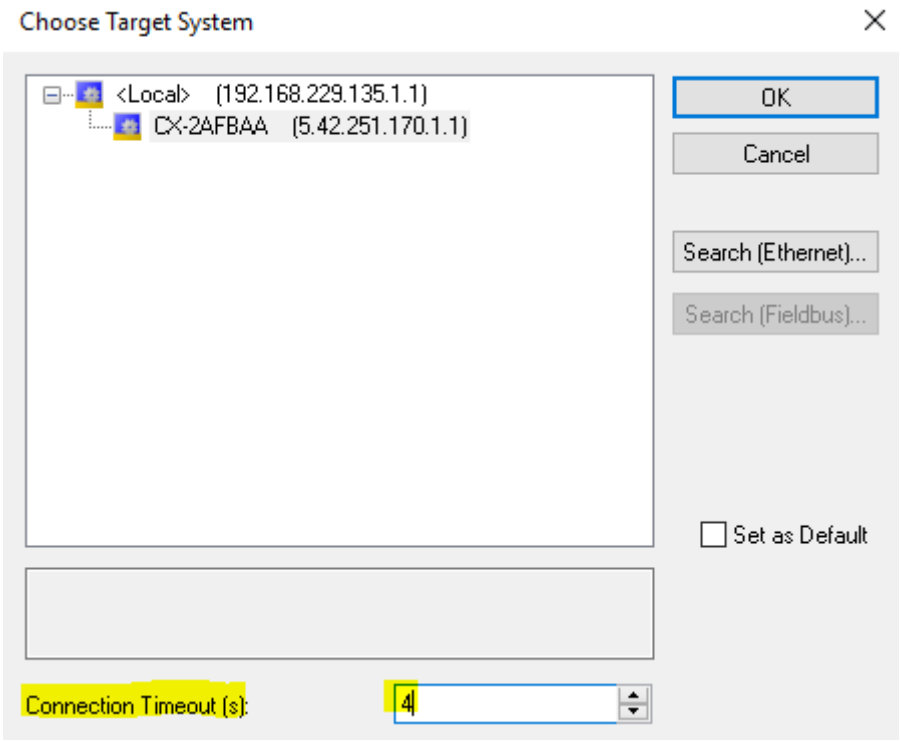
The following components provide information about the current processes in the project in offline or online mode:

Message Window	Shows current errors, warnings, and messages relating to syntax check, compile process etc.
Monitoring window and online views of editors	Used for monitoring a POU or a user-defined list of expressions/variables.
Information and status bar	Shows the status of the TwinCAT 3 runtime. If an editor window is currently active, the current cursor position and the set editing mode are displayed. In online mode you see the current program status.

Certain standard settings are installed with TwinCAT, which determine the appearance, structure and behavior of the standard components and thus the user interface. The section "Configuring user interface [▶ 33]" describes how you can edit the standard settings in order to individually adapt the user interface. Section "Reference user interface [▶ 50]" provides detailed information on the commands.

Displayed status of the TwinCAT 3 runtime

In addition to the current operation mode of the TwinCAT 3 runtime (Run mode or Config mode) the quality of the connection to the selected TwinCAT 3 runtime is displayed. The evaluation of the connection quality is always referenced to the currently set connection timeout.



The set timeout is divided into micro-timeouts and repetitions. This results in the following:

Set connection timeout = micro-timeouts * number of repetitions.

For the division between micro-timeout and repetitions, there is a case distinction:

If the set connection timeout >= 30 s,
then micro-timeout = 5 s repetitions = set connection timeout / 5

If the set connection timeout >= 10 s,
then micro-timeout = 2 s repetitions = set connection timeout / 2

Otherwise: micro-timeout = 500 ms, repetitions = set connection timeout *2

After each micro-timeout, a fresh attempt is made to communicate with the target system. No error message is output. This information is used among other things to evaluate the quality of the connection with the target system.

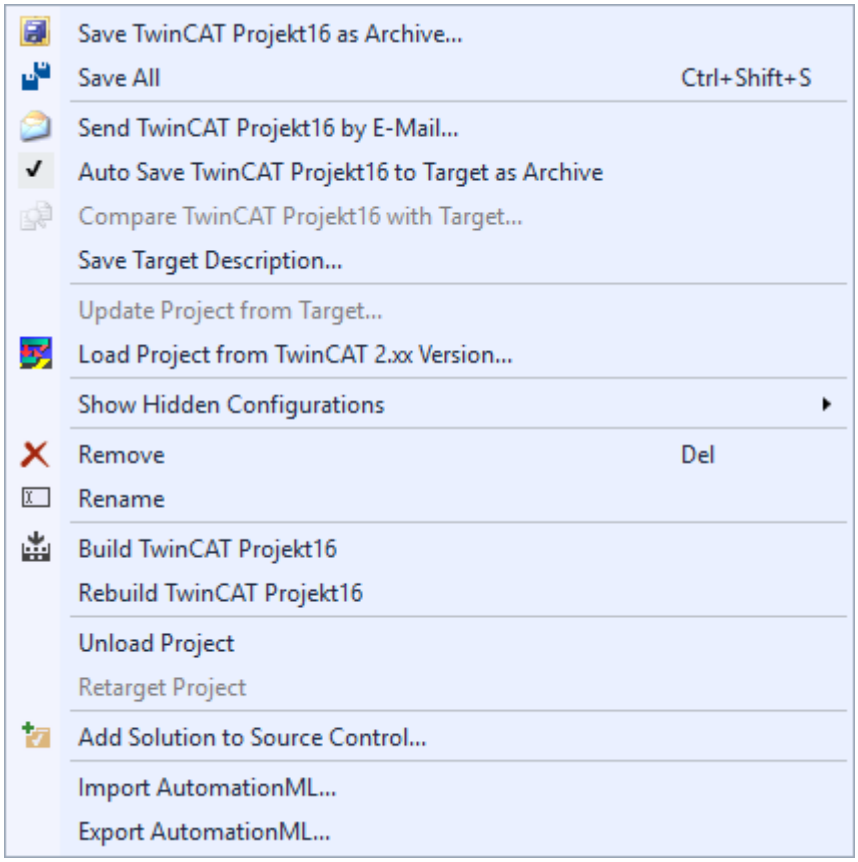
The following 3 levels exist here:

Yellow bar: 	connection quality is good: fewer than ¼ of the micro-timeouts have expired.
Orange bar: 	1st warning level: ¼ of the micro-timeouts have expired.
red bar: 	2nd warning level: half of the micro-timeouts have expired.
Flashing behavior	The display flashes in the colors explained here to indicate the micro-timeouts.

2.1 Standard menus

The menu bar is visible by default. It contains the main menus shown in the screenshot.

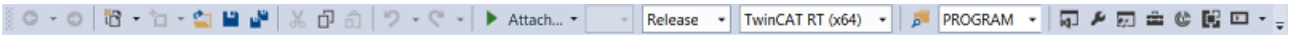
The context menu of the TwinCAT project can be called as standard.



2.2 Standard toolbars

The toolbars provide quick access to the commands. The following toolbars are set by default.

Standard Toolbar Options



TwinCAT PLC Toolbar Options

TwinCAT XAE Base Toolbar Options

2.3 Standard commands

The commands listed below are included in the main menus by default. Editor-specific menus and commands only appear when the respective editor is open. Further commands are available, which you can add to existing or newly defined menus.

File

The **File** menu contains commands for actions relating to the project file (Open, Close, Save, Print, Page settings, …)

Symbol	Command
	Command Project... (Create new TwinCAT project) [▶ 56]
	Command Project/Solution (Open Project/Solution) [▶ 51]
	Command Open Project from Target [▶ 59]
	Command Open Solution from Archive [▶ 51]
	Command New Project... (Add new TwinCAT project) [▶ 59]
	Command Existing Item... (Add existing TwinCAT project) [▶ 59]
	Command Recent Projects and Solutions [▶ 59]
	Command Save All [▶ 59]
	Command Save [▶ 60]
	Command Save <Solution name> as [▶ 60]
	Command Save <solution name> as Archive... [▶ 50]
	Command Save <TwinCAT project name> as [▶ 60]
	Command Save <TwinCAT project name> as Archive... [▶ 232]
	Command Save <PLC project name> as [▶ 60]
	Command Save <PLC project name> as archive... [▶ 52]
	Command Send by E-Mail... [▶ 61]
	Command Close Solution [▶ 61]
	Command Close [▶ 61]
	Command Exit [▶ 61]
	Command Page Setup... [▶ 62]
	Command Print [▶ 62]

Edit

The **Edit** menu contains commands for working in text editors (language editors, declaration editor…)

Symbol	Command	
	Copy	Standard Commands [▶ 62]
	Cut	
	Delete	
	Paste	
	Redo	
	Undo	
	Command Delete [▶ 63]	
	Command Select All [▶ 63]	
	Command Input Assistant [▶ 63]	
	Command Auto Declare [▶ 65]	
	Command Go To [▶ 68]	
	Command Go To Definition [▶ 68]	
	Command Navigate To [▶ 69]	
	Command Make Uppercase [▶ 69]	
	Command Make Lowercase [▶ 70]	
	Command View white spaces [▶ 70]	
	Command Quick Replace [▶ 72]	
	Command Quick Find [▶ 70]	
	Command Switch write mode [▶ 74]	
	Command Rename [▶ 74]	
	Command Rename '<variable>' [▶ 74]	
	Command Remove '<variable>' [▶ 77]	
	Command Add '<variable>' [▶ 75]	
	Command Reorder variables [▶ 78]	

View

The **View** menu contains commands for activating the individual default views in user interface windows.

Symbol	Command
	Command Open object [▶ 78]
	Command Tabular view [▶ 79]
	Command Textual view [▶ 79]
	Command Full screen [▶ 80]
	Command Toolbars [▶ 80]
	Command Solution Explorer [▶ 80]
	Command Properties Window [▶ 81]
	Command Toolbox [▶ 82]
	Command Error List [▶ 83]
	Command Output [▶ 84]

Project

The **Project** menu contains commands for handling the project objects and the project properties and for the copying and exporting projects or objects.

Symbol	Command
	Command Add New Item (project) [▶ 85]
	Command Add Existing Item (Project) [▶ 52]
	Command Add New Item (object) [▶ 243]
	Command Export to ZIP [▶ 244]
	Command Import from ZIP [▶ 245]
	Command Export PLCopenXML [▶ 245]
	Command Import PLCopenXML [▶ 245]
	Command New folder [▶ 244]
	Command Properties (PLC project) [▶ 94]
	Command Properties (object) [▶ 90]
	PLC project settings [▶ 110]

Build

The **Build** menu contains commands for creating the project through compilation (without code generation), i.e. running precompile with syntax check and commands for clearing the previous compile information, which is relevant for an Online Change and offline code generation.

Symbol	Command
	Command Build PLC project [▶ 115]
	Command Rebuild a PLC project [▶ 116]
	Command Build Solution [▶ 114]
	Command Rebuild Solution [▶ 114]
	Command Clean PLC project [▶ 116]
	Command Clean Solution [▶ 114]
	Command Check all objects [▶ 114]

Debug

The **Debug** menu contains commands for controlling the program execution on the controller (start, stop) and for debugging tasks (breakpoints, step-by-step processing, writing and forcing values).

Symbol	Command
	Command New Breakpoint [▶ 116]
	Command Edit Breakpoint [▶ 120]
	Command Enable Breakpoint [▶ 120]
	Command Disable Breakpoint [▶ 120]
	Command Toggle Breakpoint [▶ 120]
	Command Step over [▶ 121]
	Command Step into [▶ 121]
	Command Step out [▶ 121]
	Command Show Next Statement [▶ 122]
	Command Run To Cursor [▶ 122]

TwinCAT

The **TwinCAT** menu contains commands for controlling the TwinCAT 3 runtime environment.

Symbol	Command
	Command Activate configuration [▶ 123]
	Command Restart TwinCAT System [▶ 123]
	Command Restart TwinCAT (Config mode) [▶ 123]
	Command Reload Devices [▶ 123]
	Command Scan [▶ 123]
	Command Toggle Free Run State [▶ 124]
	Command Show Online Data [▶ 124]
	Command Show Sub Items [▶ 124]
	Command Hide Disabled Items [▶ 125]
	Command Choose Target System [▶ 124]
	Command Software Protection [▶ 125]
	Command Activate boot project [▶ 236]

PLC

The **PLC** menu contains commands for opening tools that provide the development environment for working with a PLC project.

Symbol	Command
	Command Breakpoints [▶ 128]
	Command Find all references [▶ 69]
	Command Cross Reference List [▶ 126]
	Command Call Stack [▶ 130]
	Command Call Tree [▶ 132]
	Command Browse Call Tree [▶ 68]
	Command Watch List <n> [▶ 125]
	Command Watch all forces [▶ 126]
	Command Add to Watch [▶ 68]
	Command Login [▶ 139]
	Command Logout [▶ 140]
	Command Download [▶ 137]
	Command Reset cold [▶ 140]
	Command Reset origin [▶ 141]
	Command Online Change [▶ 138]
	Command Start [▶ 140]
	Command Stop [▶ 140]
	Command Single cycle [▶ 141]
	Command Force values [▶ 142]
	Command Write values [▶ 144]
	Command Unforce values [▶ 143]
	Command Flow Control [▶ 141]
	Command Display Mode - Binary, Decimal, Hexadecimal [▶ 145]
	Command Create Localization Template [▶ 145]
	Command Manage Localization [▶ 145]
	Command Toggle Localization [▶ 146]
	Command Active PLC project [▶ 146]
	Command Active PLC instance [▶ 146]
	Command Memory [▶ 131]

Tools

The **Tools** menu contains commands for opening tools that create a particular environment for working with the project (installation of libraries and devices, customizing the user interface, options for editors, loading and saving etc.).

Symbol	Command
	Command Options [▶ 147]
	Command Customize [▶ 172]

Window

The **Window** menu contains commands for positioning the views in the user interface (configuration, opening, closing etc.).

Symbol	Command
	Command Float [▶ 175]
	Command Dock [▶ 175]
	Command Hide [▶ 175]
	Command Auto Hide All [▶ 176]
	Command Auto Hide [▶ 176]
	Command Pin tab [▶ 176]
	Command New Horizontal Tab Group [▶ 177]
	Command New Vertical Tab Group [▶ 177]
	Command Reset Window-Layout [▶ 177]
	Command Close All Documents [▶ 177]
	Command Window [▶ 178]
	Window submenu commands [▶ 178]

Context menu TwinCAT project

Symbol	Command
	Command Save <TwinCAT project name> as Archive... [▶ 232]
	Command Save All [▶ 59]
	Command Send <TwinCAT project name> by e-mail... [▶ 61]
	Command Backup <TwinCAT project name> automatically to the target system [▶ 233]
	Command Compare <TwinCAT project name> with the target system... [▶ 233]
	Command Update project with target system... [▶ 233]
	Command Load project with TwinCAT 2.xx Version... [▶ 233]
	Command Show Hidden Configurations [▶ 234]
	Command Remove From Solution [▶ 234]
	Command Rename [▶ 234]
	Command Build <TwinCAT project name> [▶ 115]
	Command Rebuild <TwinCAT project name> [▶ 116]
	Command Unload Project [▶ 235]
	Command Import AutomationML... [▶ 235]
	Command Export AutomationML... [▶ 235]

2.4 Default views and windows

In TwinCAT, a distinction is made between window and views.

Views

A "view" is a window within the main window of the user interface that is independent of a PLC object. Views are characterized by the following properties:

- They can be opened via the **View** menu.
- They usually contain a non-configurable toolbar with buttons for sorting, viewing and searching within the view.
- They can be docked to the frame of the main window or positioned anywhere on the screen.
- They can be hidden and represented only by a tab in the main window.

The following views are open by default:

- Solution Explorer
- Error List
- Output
- Toolbox
- Properties window

Window

A "window" is dependent on a PLC object. Windows are characterized by the following properties:

- They can be opened via the **Window** menu or by double-clicking on the object.
- They are all arranged in a particular area of the user interface as MDI windows in the form of tabs. The arrangement of the windows depends on the user-specific settings.
- They cannot be hidden.
- They are not open by default.

Examples:

- Editor window for PLC objects

2.5 Information bar

The information bar shows the current cursor position and the edit mode as well as the mode of the TwinCAT 3 Runtime, depending on which view or window is active.

Cursor position

The current cursor position is displayed when an editor window is active.

Current cursor position, starting from the left or top of the editor window:

L (Ln)	Line number
C (Col)	Column number (a column contains exactly one space, letter or digit)
Chr (Ch)	Number of characters (a character may refer to a single letter, a single digit, or even a tab range, which may cover 4 columns, for example)

Double-clicking on one of the fields opens the **Go To Line** dialog. Here you can specify a line in which the cursor should be placed.

Edit mode

The current editing mode is displayed when an editor window is active.

Currently set editing mode:

INS	insert mode
OVR	overwrite mode

Double-click the field to change the setting.

TwinCAT system state

You can recognize the state of the TwinCAT system by the color of the TwinCAT system service icon in the system tray. The following states are possible:

	TwinCAT 3 Runtime is in Config mode
	TwinCAT 3 Runtime is in Run mode
	TwinCAT 3 Runtime is in Stop mode
	TwinCAT 3 Runtime is in Exception mode
	Flashing behavior indicates the micro-timeouts [▶ 15].

2.6 Status symbols

Status of the PLC modules

PLC object	PLC instance	Status
		TwinCAT in config mode PLC logged out PLC stopped
		TwinCAT in Run mode PLC logged in PLC stopped
		TwinCAT in Run mode PLC logged in PLC started
		TwinCAT in Run mode PLC logged out PLC stopped
		TwinCAT in Run mode PLC logged out PLC started
		TwinCAT in Run mode Another user is logged into the PLC. PLC started

Status of PLC objects

	The object has not changed since it was last saved.
	The object has changed since it was last saved.
	The object is signed.
	The object is encrypted.

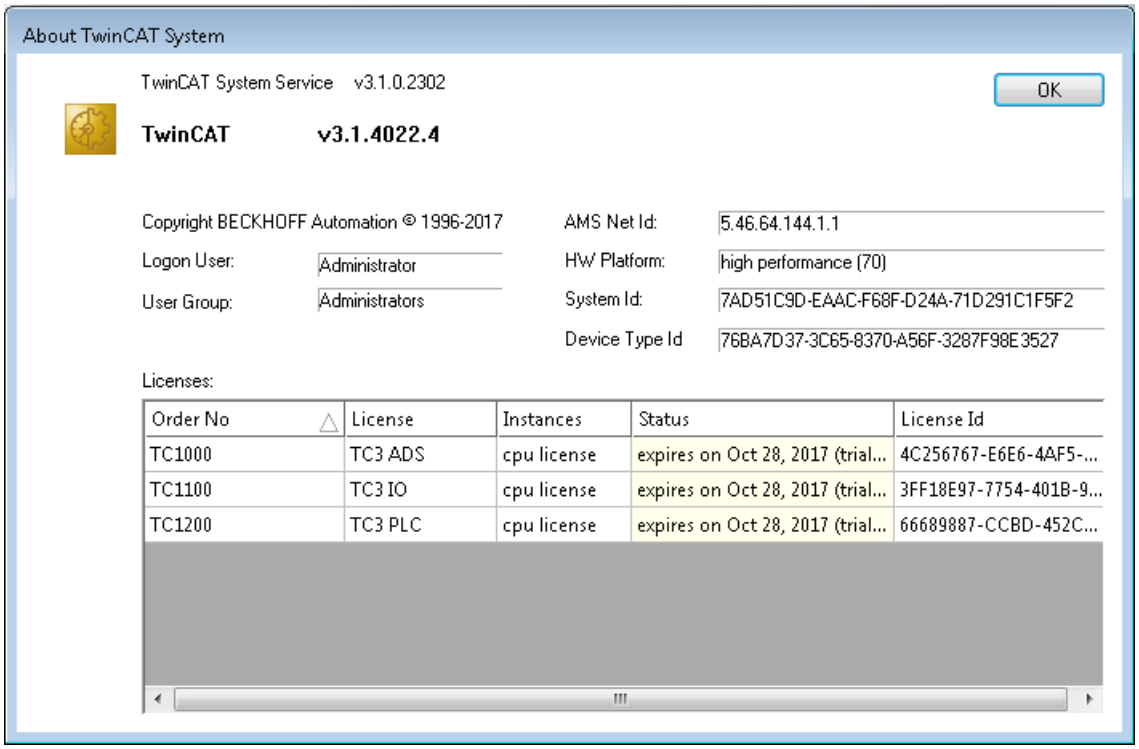
2.7 System menu

The TwinCAT system menu provides all the important commands for access to functions of the TwinCAT system. In addition to important runtime commands for starting and stopping the runtime, the menu contains commands for accessing the router settings and the communication between engineering and runtime. The router commands are especially important when engineering and runtime are on separate computers. The engineering environment can also be started directly via the TwinCAT system menu.

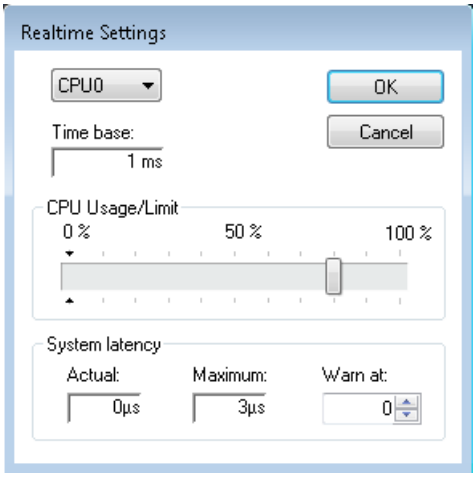
Open the system menu by right-clicking on the TwinCAT system service icon in the system tray.

2.7.1 About TwinCAT…

The **TwinCAT System** dialog lists all the essential data of the installed TwinCAT version. This can be copied to the clipboard or emailed to the TwinCAT support hotline, as required.



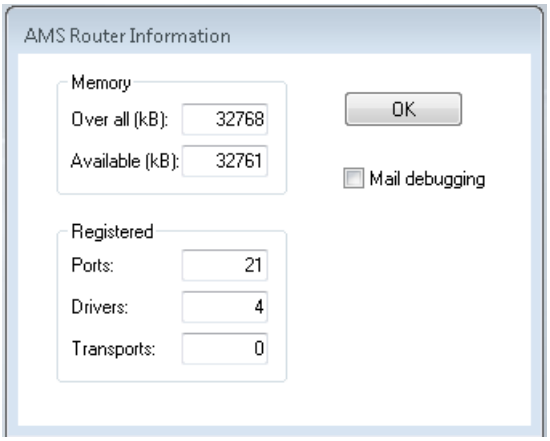
2.7.2 Realtime



Time base	Timebase for calculating the percentage. Currently, a fixed timebase of 1 millisecond is set.
CPU Usage/Limit	The slider can be used to allocate processor time to the TwinCAT real-time system. Currently, a timebase of 1 millisecond is set. In the example shown, TwinCAT is allocated a maximum of 80 % of the computing time. With a timebase of 1 ms, this means that TwinCAT has a maximum of 800 µs available per millisecond. It also means that at least 200 µs remain for Windows. When the TwinCAT real-time system switches to its idle task, the processor is returned to Windows. The slider bar shows the current utilization of the real-time system. The display is averaged over 256 cycles (ms).
System latency	This shows the current and maximum latency time in the real-time system. The time by which the central system tick is delayed is measured. The maximum time is stored until the slider is operated or the dialog is exited. The latency time is, of course, also measured when the dialog is not open. If the set maximum time is exceeded, a message is displayed once in the window, and a logbook entry is created. By calling up the dialog, the message can be reset so that a message is displayed again when the next timeout occurs.

2.7.3 Router

Info



Memory

The displayed RAM is required in the TwinCAT system for AMS messages and for memory management in the TwinCAT real-time environment. The entire memory is requested by Windows when the TwinCAT system starts up. The memory size can be configured in the TwinCAT system node.

Over all	RAM that was available when the TwinCAT system was started.
Available	RAM that is currently available for TwinCAT.

Registered

All devices taking part in the TwinCAT messaging system (AMS) have to register with the router. TwinCAT servers have fixed port numbers (e.g. TwinCAT PLC LZS1: 851, …). TwinCAT clients are assigned a port number by the message router.

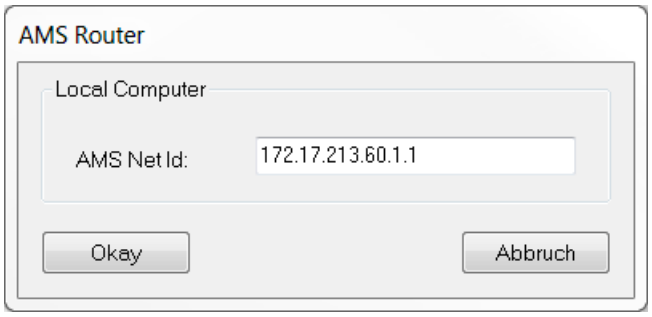
Ports	Number of registered ports
Drivers	Ports occupied by TwinCAT servers
Transports	Number of registered devices (NetIDs)

Mail debugging	In conjunction with the TwinCAT ADS Monitor, the whole TwinCAT message traffic can be logged. Note that the message traffic is slowed down by the additional messages. The TwinCAT ADS Monitor is not part of the standard scope of supply of TwinCAT.

Cleanup

The cleanup function of the router can be used to release router ports originating from programs that are no longer functional. This function is particularly useful in the development phase of PLC projects.

Changing the AMS NetID



Local computer

AMS NetID	The AMS Net ID is the address of the local computer in the TwinCAT network. The AMS Net ID consists of 6 bytes and is represented in a dot notation. The “Net IDs” must be assigned by the project planner and must not be repeated in the TwinCAT network. By default, the installation generates an AMS NetID from the IP address of the system (if available) + “1.1”. If no IP address can be determined during the installation, the AMS NetID “1.1.1.1.1.1” is created.

Editing routes

The **TwinCAT Static Routes** dialog shows all the route information of the local TwinCAT system.

The **Add** button can be used to add new routes.

2.7.4 System

If the TwinCAT system is not configured for automatic startup, you can start it manually via the system menu.

Start/Restart	The TwinCAT system is started. All TwinCAT servers that were entered are loaded and initialized. The TwinCAT I/O subsystem is parameterized by the TwinCAT I/O Manager, according to the configuration. All runtime systems of the PLC subsystem that were entered are initialized. If a boot project is entered for a runtime system, it is loaded and the PLC program is started. The remanent (retain) data is also loaded according to the configuration.
Config	The TwinCAT system is stopped. All TwinCAT servers that were entered are shut down and unloaded. Once the TwinCAT system has stopped, only the TwinCAT Message Router remains in the memory. The TwinCAT system can now be started again via System > Start.

2.7.5 Tools

Tool		Description
Event Viewer		Opens the Windows Event Viewer, in which system errors are entered
TwinCAT Project Compare		Opens an application program for comparing or merging TwinCAT projects See: Documentation Source Control > Introduction
TwinCAT Switch Runtime		Opens an application program for switching the TwinCAT runtime environment between TwinCAT 2 and TwinCAT 3

2.8 User interface in online mode

As soon as you log on to the target system with a project, all objects that were opened in windows and views in offline mode are displayed in online mode.

For further information on the online views of the individual editors see the description of the respective editor.

See also:

- PLC documentation: SFC editor in online mode
- PLC documentation: CFC editor in online mode
- PLC documentation: FBD/LD/IL editor in online mode
- PLC documentation: ST editor in online mode

3 Configuring user interface

You can customize TwinCAT's behavior, appearance, menu composition and window layout. In the **Tools** menu you will find dialogs for customizing the interface and setting the TwinCAT options.

See also:

- Command Options [▶ 147]
- Command Customize [▶ 172]

3.1 Setting options

You can configure the behavior and appearance of TwinCAT in the various tabs of the **Options** dialog. The dialog opens after selecting the **Options** command in the **Tools** menu. Here you can implement the settings for various editors and functionalities. These settings are valid across TwinCAT. The settings are stored in your current user profile on your local system.

See also:

- Command Options [▶ 147]
- Customizing the user interface [▶ 34]

3.1.1 Customizing text editor settings

You should use identical font and editor settings, in order to achieve a uniform typeface in different programs. You can customize these settings in the TwinCAT options.

When TwinCAT 3 is integrated into Visual Studio, the font and tabulator size are preset by default.

Customizing the font and colors

1. Select the **Options** command in the **Tools** menu.
   ⇨ The **Options** dialog opens.
2. Select the category **Environment > Fonts and colors**.
3. Select the desired font settings.
4. Close the dialog with **OK**.

⇨ The settings are applied after restarting the program.

Default settings:

Element	Text only
Font	Consolas
Size	10
Item foreground	Standard (black)
Item background	Default (white)

Customizing the text layout

1. Select the **Options** command in the **Tools** menu.
   ⇨ The **Options** dialog opens.
2. Select the category **TwinCAT > PLC Environment > Text editor**.
3. Select the desired layout settings.
4. Close the dialog with **OK**.

⇨ The settings are applied after restarting the program.

Default settings:

Number of undo steps	100
Folding	Indent
Word wrap	None
Tab width	4
Keep tabs	Yes
Number of indent characters	4
Auto Indent	Smart with code completion

3.1.2 Select language

1. Select the **Options** command in the **Tools** menu.
   ⇨ The **Options** dialog opens.
2. Select the category **Environment> International settings**.
3. Select a language.

⇨ The settings are applied after restarting the program.

3.2 Customizing the user interface

TwinCAT offers the option to customize the user interface to your needs. You can customize the layout of the windows as well as the visibility of menus and commands.

3.2.1 Customizing Menus

You can customize the menu commands of the user interface to suit your needs. You can use the **Customize** dialog to hide or add menus.

Restoring the default settings

Use the **Reset all** button under **Tools > Customize > Commands** to reset the changes you made in the customization settings.

Special menu bars

Some views, for example in the Solution Explorer, have a special menu bar with buttons for sorting, displaying and searching in the window. These menu bars cannot be configured.

Removing a menu or command

1. Select the **Customize** command in the **Tools** menu.
   ⇨ The **Customize** dialog opens. The **Toolbars** tab is visible.
2. Select the **Commands** tab.
   ⇨ The **Menu bar** field is activated by default.
3. In the corresponding listbox, select the menu you want to edit.
   ⇨ The **Commands** area shows the available submenus and commands.
4. Select a menu or command.
5. Click **Delete**.
   ⇨ The menu or the command is deleted from the menu tree.
6. Click **Close**.

⇨ The dialog closes, and the menu is customized.

Adding a menu

1. Select the **Customize** command in the **Tools** menu.

⇨ The **Customize** dialog opens. The **Toolbars** tab is visible.
2. Select the **Commands** tab.
⇨ The **Menu bar** field is activated by default.
3. In the corresponding listbox, select the menu you want to edit.
⇨ The **Controls** area shows the available submenus and commands.
4. Click **Add New Menu**.
⇨ A new menu is added.
5. Click on the **Modify Selection** combobox and enter a name for the new menu.
6. Change the position in the menu sequence using the **Move Up** or **Move Down** buttons.
7. Click **Close**.

⇨ The dialog closes. The new menu is not fully displayed until a command is available within the menu.

Adding a command

1. Select the **Customize** command in the **Tools** menu.
⇨ The **Customize** dialog opens. The **Toolbars** tab is visible.
2. Select the **Commands** tab.
⇨ The **Menu bar** field is activated by default.
3. In the corresponding listbox, select the menu you want to edit.
⇨ The **Commands** area shows the available submenus and commands.
4. Click **Add Command…**.
⇨ The **Add Command** dialog opens. The dialog lists all commands grouped into categories.
5. Select the command you want to add and click **OK**.
⇨ The new command is added to the menu tree.
6. Change the position in the menu sequence using the **Move Up** or **Move Down** buttons.
7. Click **Close**.

⇨ The dialog closes, and the new command is available in the menu.

See also:

- Command Customize [▶ 172]

3.2.2 Customizing Toolbars

You can customize the toolbars of the TwinCAT user interface to suit your needs. You can use the **Customize** dialog to hide or add toolbars.

Showing or hiding a toolbar

1. Select the **Customize** command in the **Tools** menu.
⇨ The **Customize** dialog opens. The Toolbars tab is visible.
2. Select the check boxes for the toolbars you want to show or hide.
⇨ The toolbar is displayed or hidden in the user interface.
3. Click **Close**.

⇨ The dialog closes.

Adding and removing a toolbar

1. Select the **Customize** command in the **Tools** menu.
⇨ The **Customize** dialog opens. The Toolbars tab is visible.
2. Click **New**.
⇨ The **New Toolbar** dialog opens.

3. Enter a name and click **OK**.
   ⇨ A new, empty toolbar is automatically activated and added below the menu bar.
4. Click **Close**.

⇨ The dialog closes. The new toolbar is visible. The new toolbar does not become fully visible until a command is available within the toolbar.

Use the **Delete** command in the **Customize > Toolbars** tab to remove a toolbar you have compiled.

Adding and removing a command

1. Select the **Customize** command in the **Tools** menu.
   ⇨ The **Customize** dialog opens. The **Toolbars** tab is visible.
2. Select the **Commands** tab.
   ⇨ The **Menu bar** field is activated by default.
3. Activate the Toolbar field and select the menu you want to edit from the corresponding listbox.
4. Click **Add Command…**.
   ⇨ The **Add Command** dialog opens. The dialog lists all commands grouped into categories.
5. Select the command you want to add and click OK.
   ⇨ The new command is added to the toolbar.
6. Change the position in the menu sequence using the **Move Up** or **Move Down** buttons.
7. Click **Close**.

⇨ The dialog closes. The new command is available in the toolbar.

Use the **Delete** command in the **Customize > Commands** tab to remove a command from the toolbar.

See also:

- Command Customize [▶ 172]

3.2.3 Customizing Keyboard Shortcuts

TwinCAT offers you the option of calling up commands directly via a hotkey. You can customize or extend the predefined hotkeys.

1. Select the **Customize** command in the **Tools** menu.
   ⇨ The **Customize** dialog opens. The **Toolbars** tab is visible.
2. Select the **Commands** tab.
3. Click **Keyboard…**
   ⇨ The **Options** dialog opens.
4. Select the command in the listbox **Show commands containing**.
5. Enter a hotkey in the text field **Press shortcut keys**.
6. Click **Assign**.
   ⇨ The hotkey is assigned to the command.
7. Click on **OK**.

⇨ You can now call the command using the hotkey.

See also:

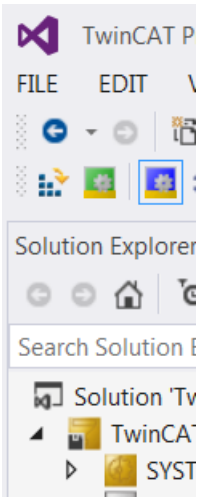
- Command Customize [▶ 172]

3.2.4 Arranging the menu bar and the toolbar

The menu bar is always located at the top of the main user interface window, between the title bar and the window area.

A toolbar can be docked below the menu bar, or it can be positioned as an independent window anywhere on the screen.

To move a bar, click with the mouse pointer on the dots at the left end of the bar, hold down the mouse button and move the bar to the desired position.

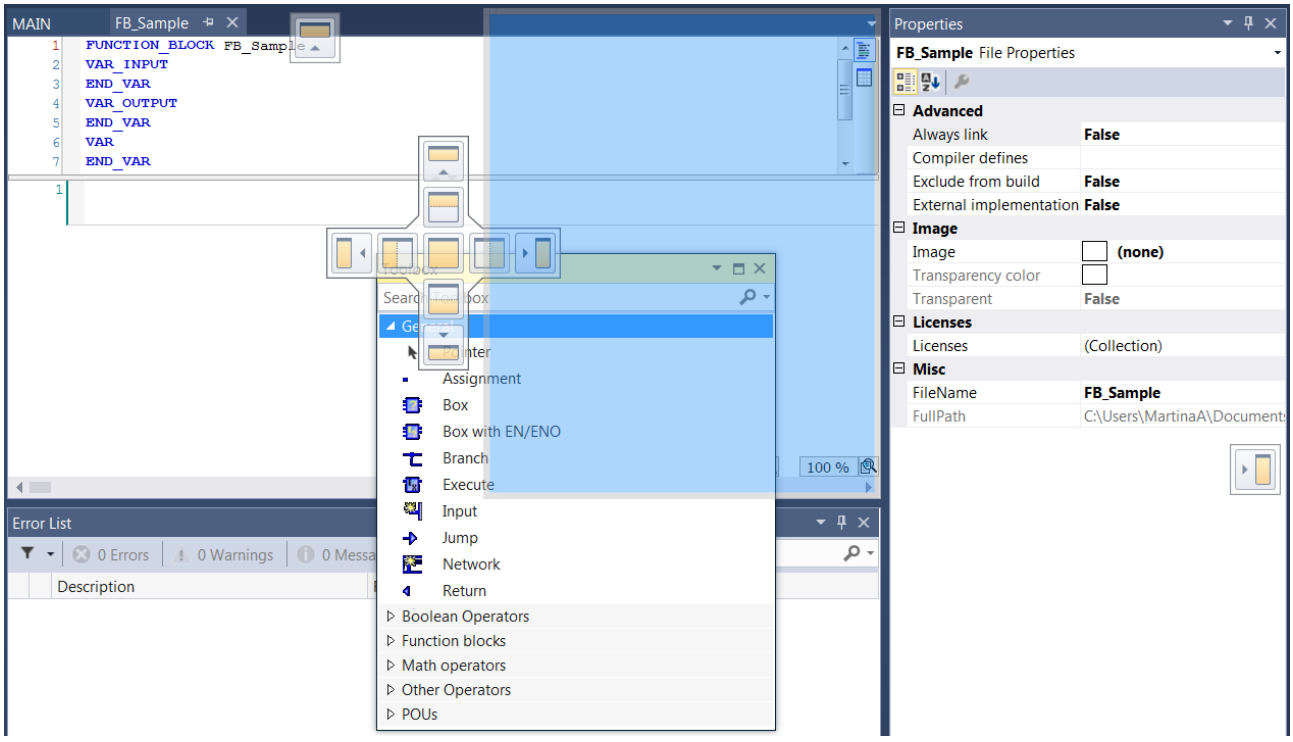


3.2.5 Arranging views and windows

In TwinCAT you can easily adjust the arrangement of the different windows to suit your individual needs.

1. Drag the window on the title bar or tab.
   ⇨ Arrow icons are displayed, which indicate the possible positions.
2. Drag the window with the mouse onto one of the arrow symbols.
   ⇨ The target position is displayed as a blue shaded area.
3. Release the left mouse button.

⇨ The window is inserted in the selected target area.



If required, you can position the window outside the TwinCAT programming interface.

See also:

- Toggle between views and windows [▶ 38]
- Show/hide views [▶ 38]
- Resizing Windows [▶ 38]

3.2.6 Toggle between views and windows

You can toggle directly between the currently open views and editor windows.

1. Press the **[Ctrl]** and **[Tab]** keys simultaneously. Continue to hold down the **[Ctrl]** key.
   ⇨ An overview of all active views and editors opens.
2. Press and hold the **[Ctrl]** key while using the arrow keys to select the window.
3. Release the **[Ctrl]** key.

⇨ The selected view or the selected editor is activated.

3.2.7 Show/hide views

Hide views

If you hide a view, it is only available as a tab within the user interface. When you click the tab, the view will appear automatically.

1. Click in the view you want to hide.
2. Activate the **Auto Hide** command in the **Window** menu or click the  button in the upper right corner of the view.
   ⇨ The view is hidden and indicated only by a small tab at the side of the main window.
3. Click the tab.

⇨ The view is displayed.

The **Hide** command on the **Window** menu closes the view.

Showing views

1. Click on the tab of the hidden view.
2. Select the **Dock** command in the **Window** menu or click the  button in the upper right corner of the view.

⇨ The view is permanently displayed.

See also:

- Command Auto Hide [▶ 176]
- Command Hide [▶ 175]
- Command Dock [▶ 175]

3.2.8 Resizing Windows

1. Move the mouse over the dividing line between two windows or views.
   ⇨ The cursor becomes a two-ended arrow.
2. Use the mouse to drag the separator line to the desired position.

⇨ While one window gets bigger, the other window gets smaller.

You can resize detached windows by moving the border line.

4 Use TwinCAT Documentation

4.1 Calling the TwinCAT 3 documentations

From Visual Studio 2013, the TwinCAT 3 help system uses the Beckhoff Information System if no local TwinCAT Information System has been installed. In this case, the TwinCAT 3 context help uses the browser of the operating system to search for the articles online in the Beckhoff Information System and to display them according to the search in the system's web browser.

An Internet connection is required for this and the address https://infosys.beckhoff.com must be reachable.

With the installation of the TwinCAT 3 Information System, the help system is set to use the local help.

The use of the local help or the online help respectively can be set in the TwinCAT Help menu (Visual Studio 2013 and later).

In the Help menu, open the submenu "Define help settings".

- **Browser**: Searches online in the Beckhoff Information System for articles on the context help used.
- **Help Viewer**: Searches in the installed local TwinCAT 3 Help for the corresponding articles

You can call the local help system via the Windows Start menu or in the TwinCAT 3 Engineering (XAE) via the context help (F1 Help) or the **Help** menu.

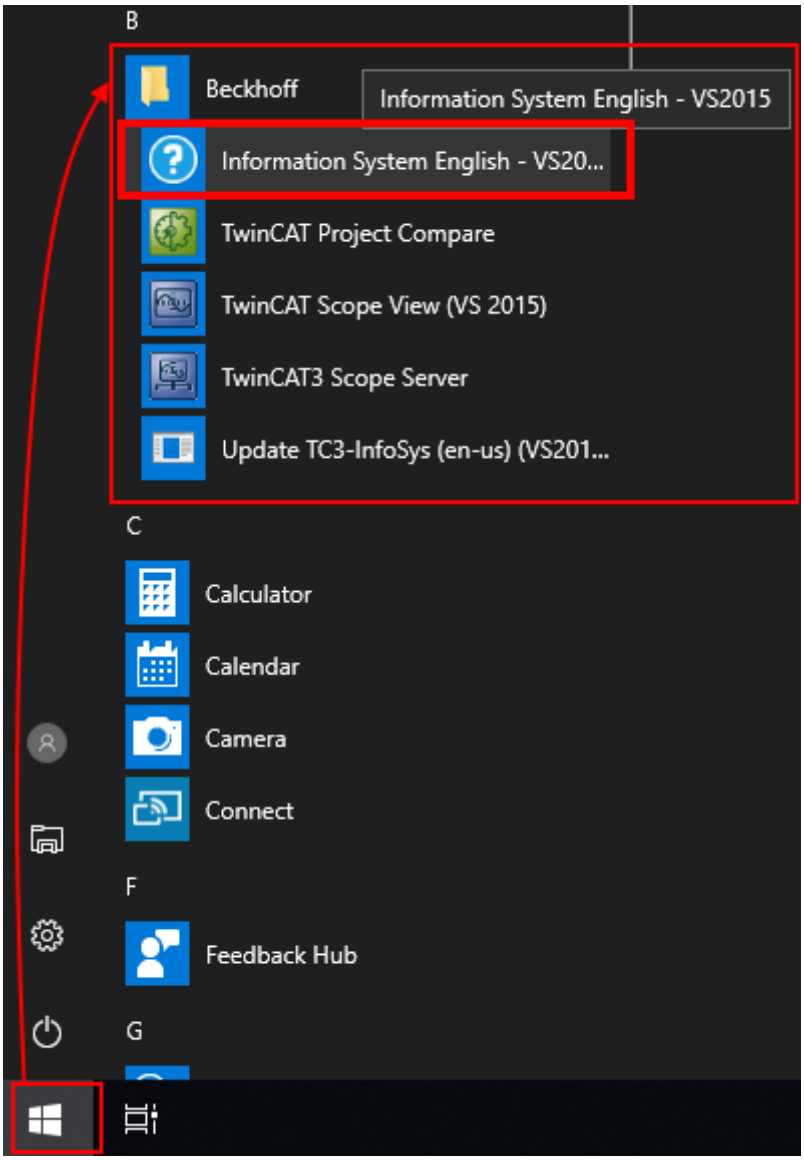
- Calling up the TwinCAT 3 documentation via the Windows Start menu [▶ 42]
- Calling up the TwinCAT 3 documentation via the Help menu [▶ 41]
- Calling up the TwinCAT 3 documentation via the context help [▶ 42]

Calling up the TwinCAT 3 documentation via the Windows Start menu

✓ You have downloaded and installed the TwinCAT 3 Information System.

✓ During the installation, a shortcut to the Help Viewer was created in the Windows Start menu. Depending on the system configuration and the (pre-)set installation components, several links may be available (e.g. for the German and English versions of the TwinCAT 3 Information System).

1. Open the **Beckhoff** folder in the Windows Start menu.



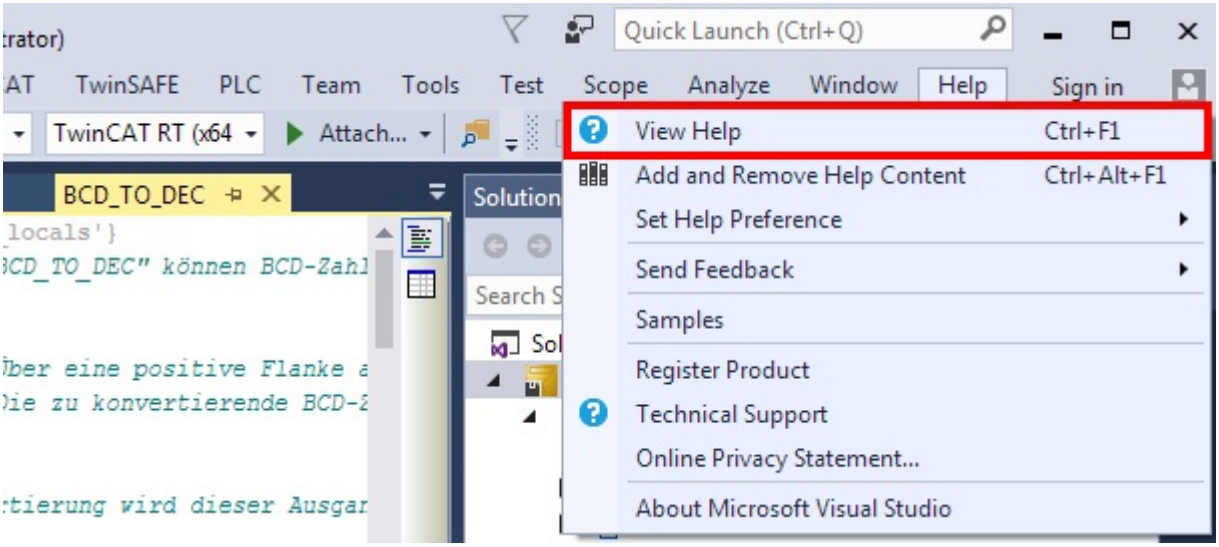
2. Click **Information System <Language> - VS <Version>**.

⇨ The TwinCAT 3 Information System is opened with Help Viewer.

Calling up the TwinCAT 3 documentation via the Help menu

✓ TwinCAT 3 Engineering is open.

1. Open the **Help** menu and select the **Show Help** command.



⇨ The TwinCAT 3 Information System is opened with the Help Viewer of the Visual Studio help system. In the tree view of the open Help Viewer, select **Help Viewer - Home > Welcome to the Beckhoff Information System > TwinCAT 3**.

Calling up the TwinCAT 3 documentation via the context help

With the context help, the position of the mouse pointer (for graphical elements) or a selected text expression within the editor is evaluated by the help system in order to display the appropriate linked help text.

Limited help function due to different language settings

If you have installed the TwinCAT 3 Information System in a language other than Visual Studio, the help system is only available to a limited extent.

Various displays

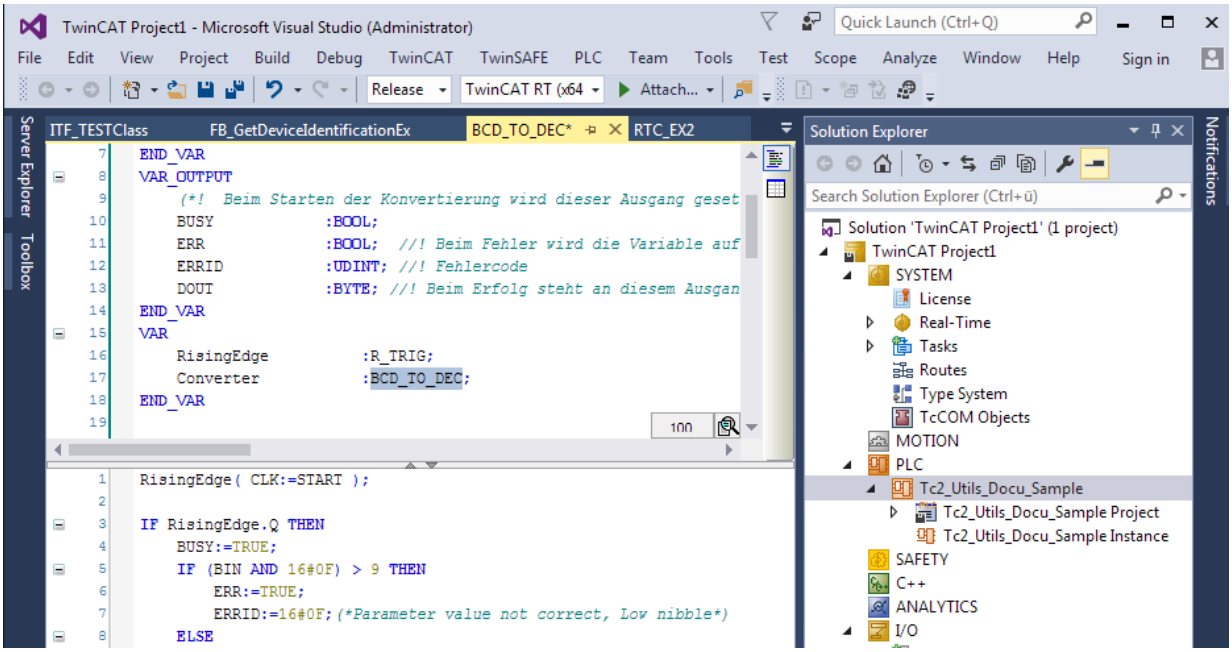
From Visual Studio 2013 it is possible to use the context help with the Beckhoff Information System. Depending on the setting of the help system [▶ 40], the result is displayed locally in the Help Viewer or in the system's web browser.

The following guide shows you how to use the local TwinCAT 3 Help. The online help is used analogously. Only the articles are displayed in the web browser.

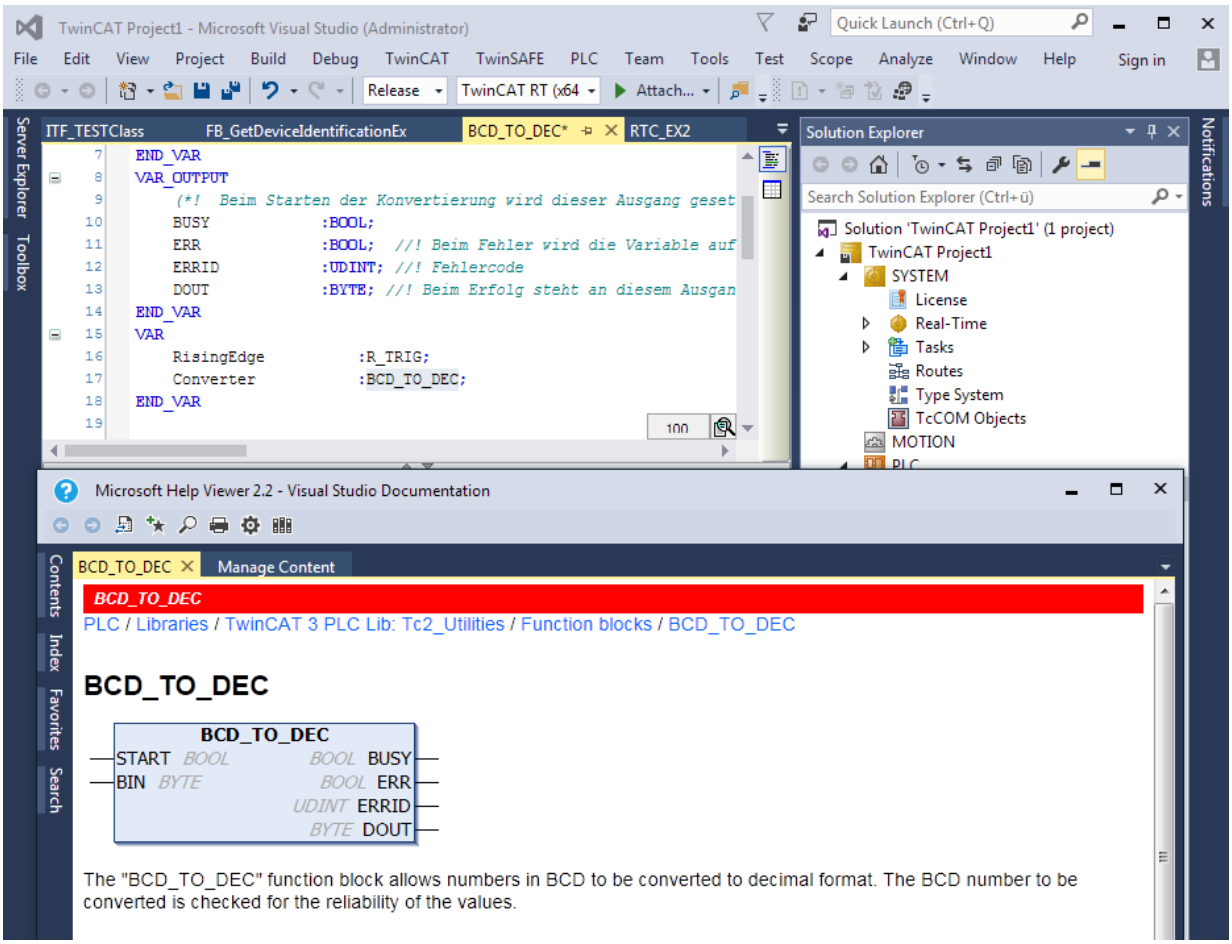
✓ A TwinCAT 3 project is opened.

✓ The **Start in Help Viewer** command is enabled in the menu **Help > Define help settings** (from Help Viewer 2.0).

1. Open an editor for the supported text-based TwinCAT 3 programming languages, e.g. ST or C++.
2. Select the desired expression and press the **[F1]** key.



⇨ The Help Viewer opens, and the corresponding article from the TwinCAT 3 Information System linked to the selected expression is displayed.



4.2 Updating the TwinCAT 3 documentation

All language variants of the TwinCAT 3 Information System are updated weekly. A locally installed TwinCAT 3 Information System can be updated manually via the Visual Studio help system. The procedure depends on the Visual Studio version or the help system.

See also:

- Installation of the TwinCAT 3 documentation

4.2.1 Update in Visual Studio

The procedure described below for updating the TwinCAT 3 Information System is valid for Visual Studio 2012 / 2013 / 2015 / 2017 / 2019.

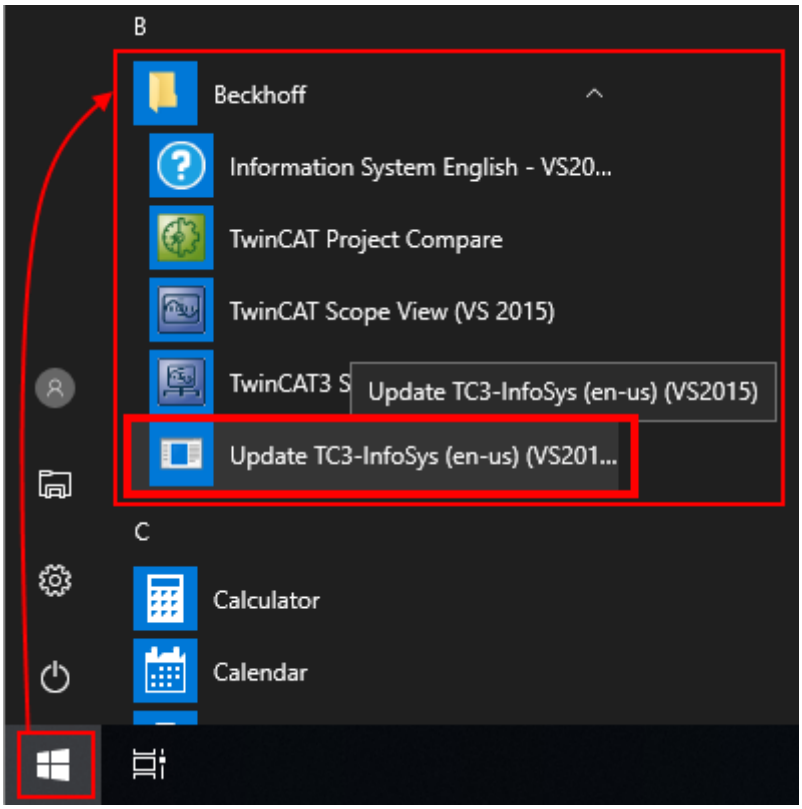
When the TwinCAT 3 Information System is installed, an entry is created in the Windows start menu, which can be used to update the TwinCAT 3 Information System directly.

✓ An internet connection is available.

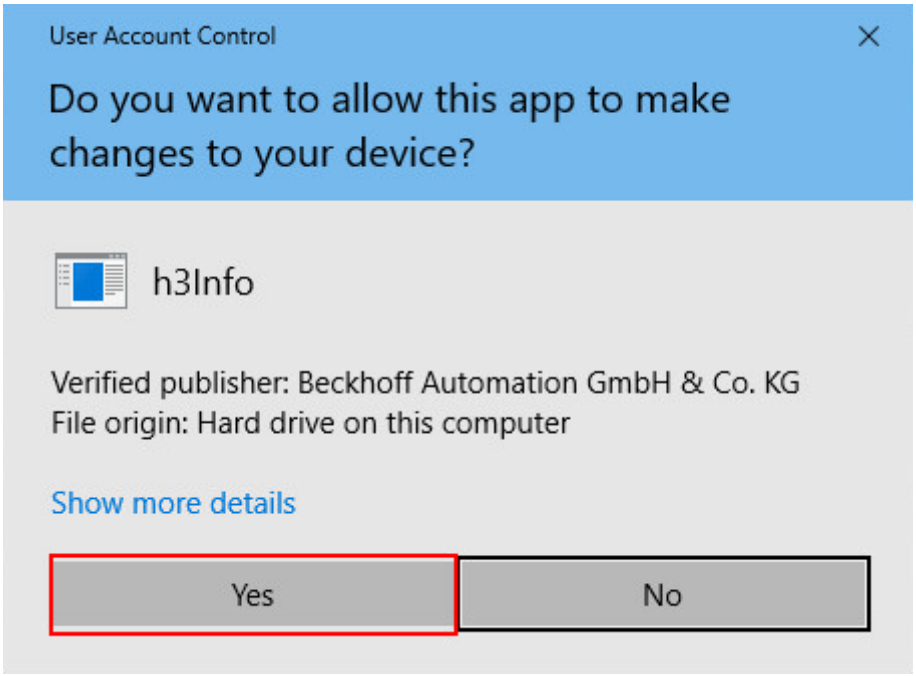
✓ You have administrator rights.

1. Open the **Beckhoff** folder in the Windows Start menu.

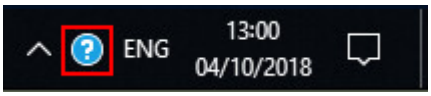
2. Click **Update TC3-InfoSys (<Language>) (VS<Version>)**.



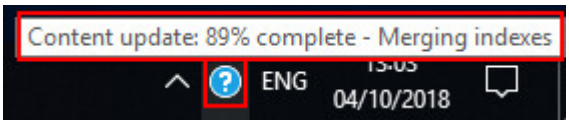
3. In the dialog that opens, confirm that changes may be made to the system.



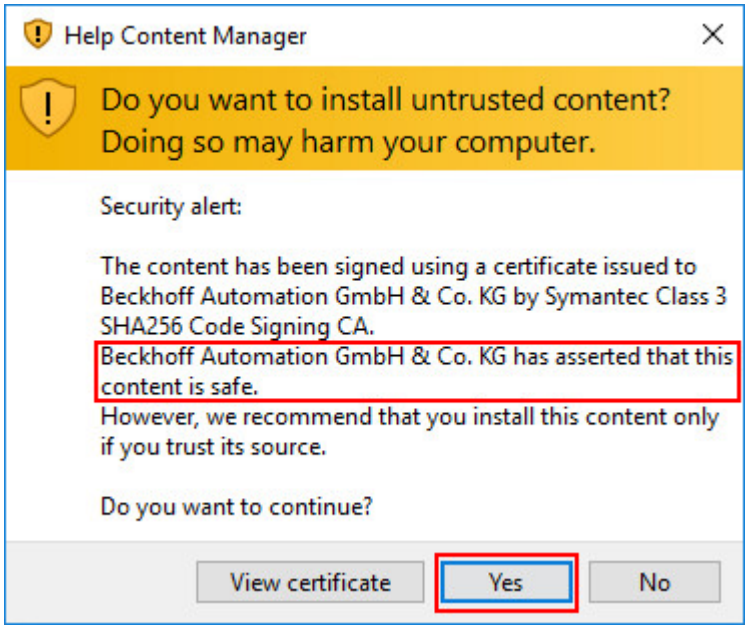
⇨ The help system checks whether an update for the TwinCAT 3 Information System is available. A help system icon is displayed in the Windows menu bar.



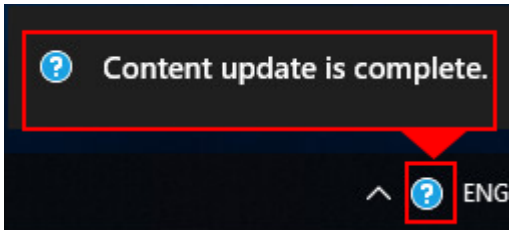
When you move the mouse pointer over the help system icon, the update progress is displayed.



4. Confirm that the TwinCAT 3 Information System may be updated.



⇨ The update status is displayed in the Windows menu bar:



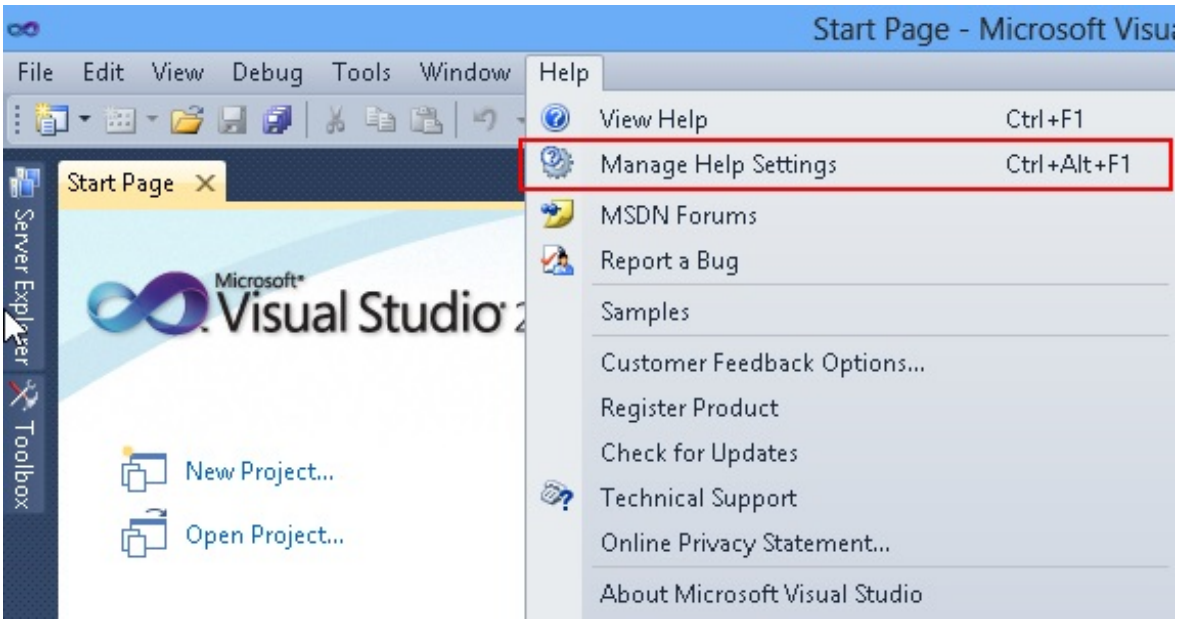
4.2.2 Update in Visual Studio 2010

If an internet connection is available, the installed TwinCAT 3 Information System can be updated directly via the Visual Studio help system in TwinCAT 3 Engineering.

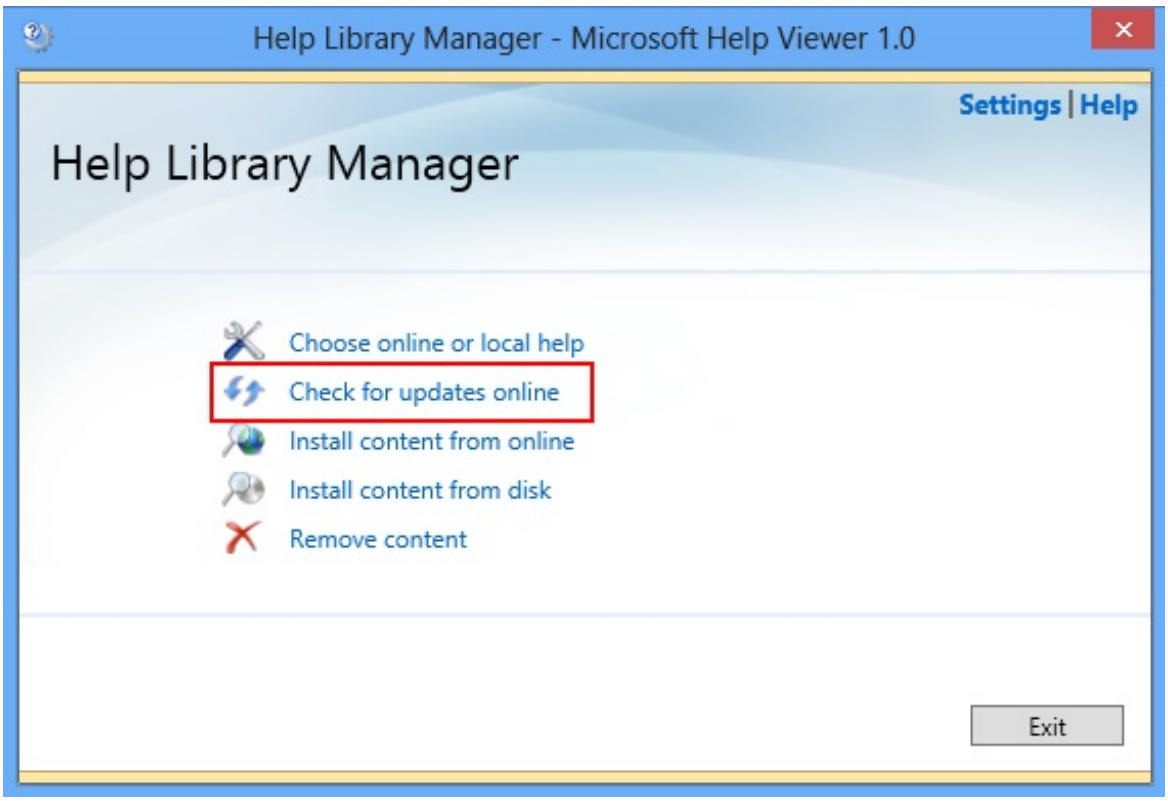
✓ An internet connection is available.

1. Start TwinCAT 3 Engineering or Visual Studio 2010.

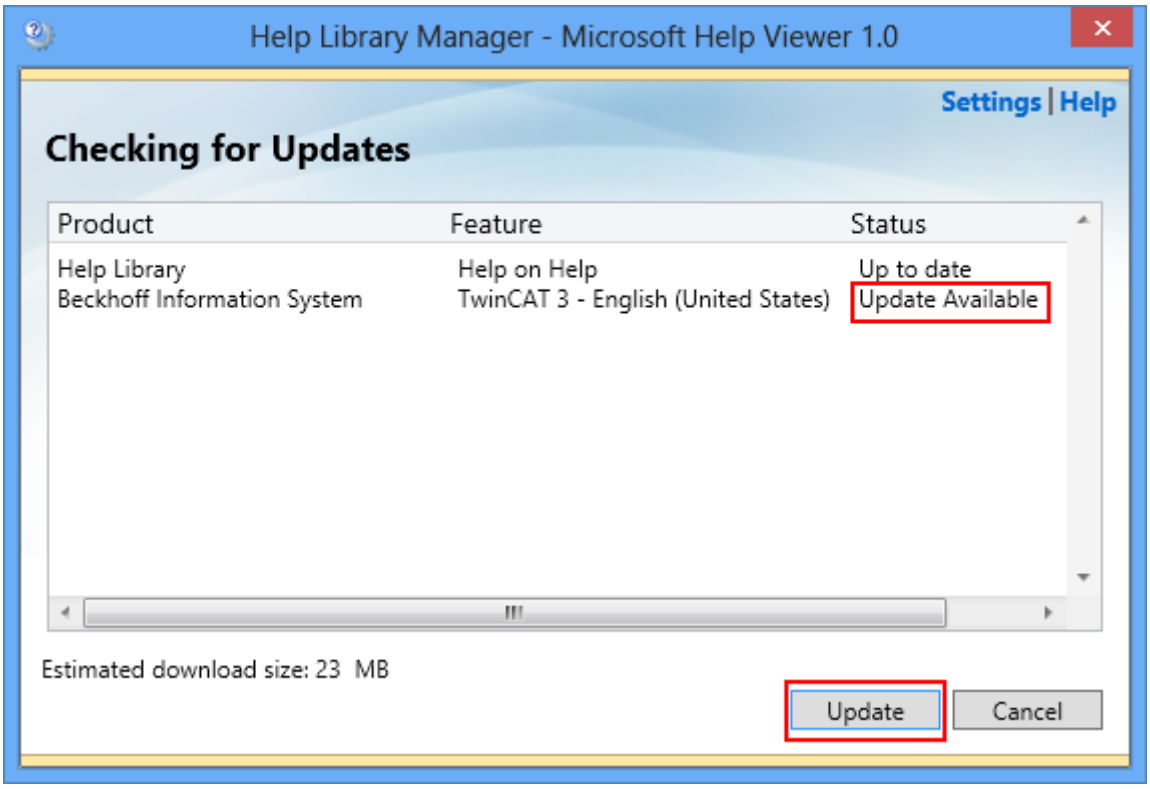
2. Select **Manage Help Settings** from the **Help** menu.



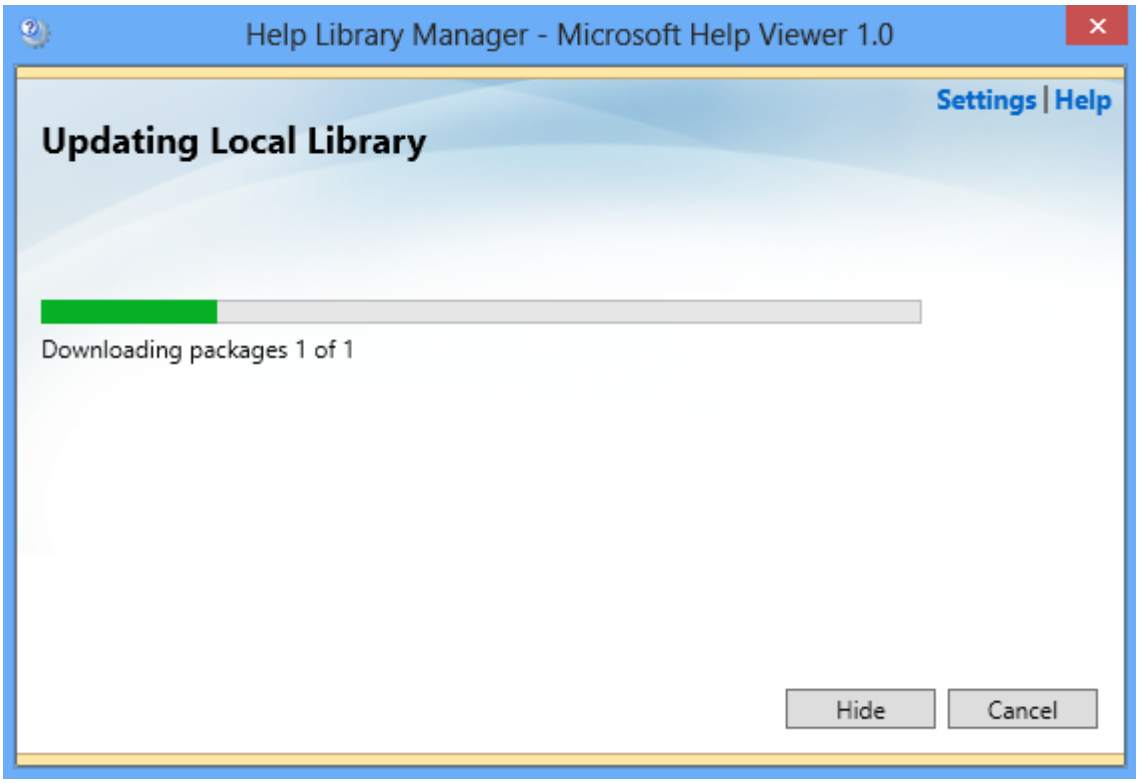
3. In the help system dialog that opens, click **Check for updates online**.



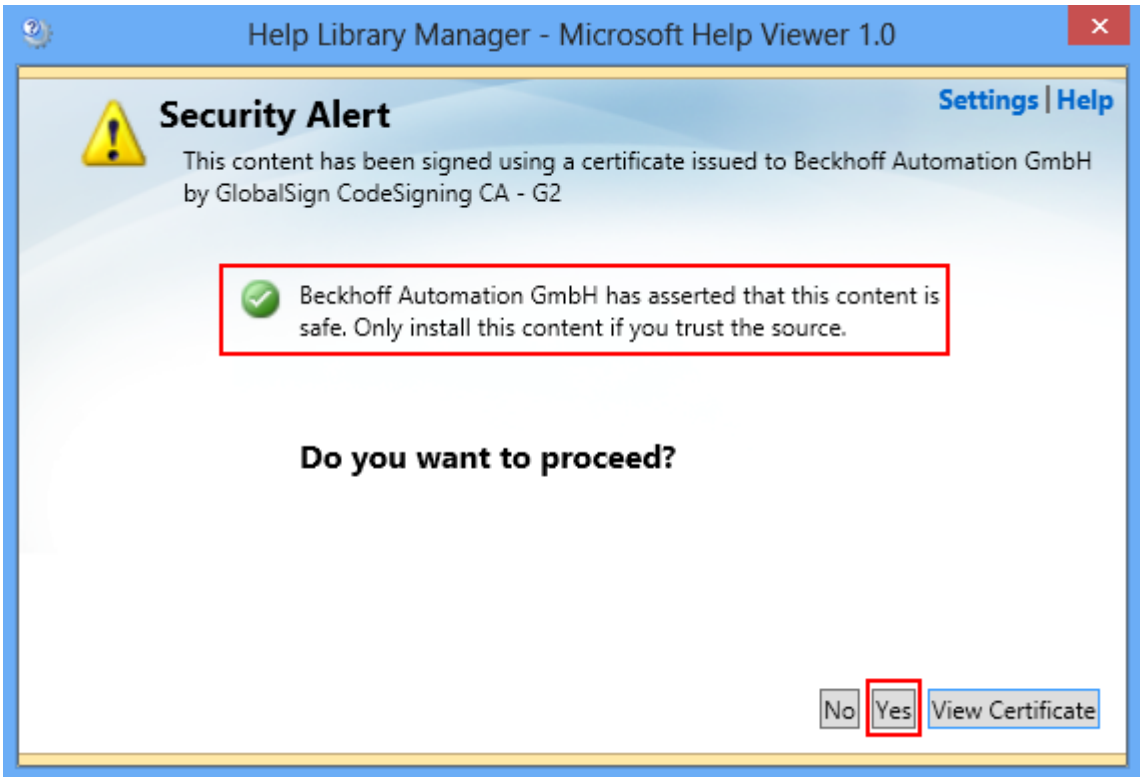
⇨ The help system checks whether an update for the TwinCAT 3 Information System is available. Available updates for installed documentation components are displayed.



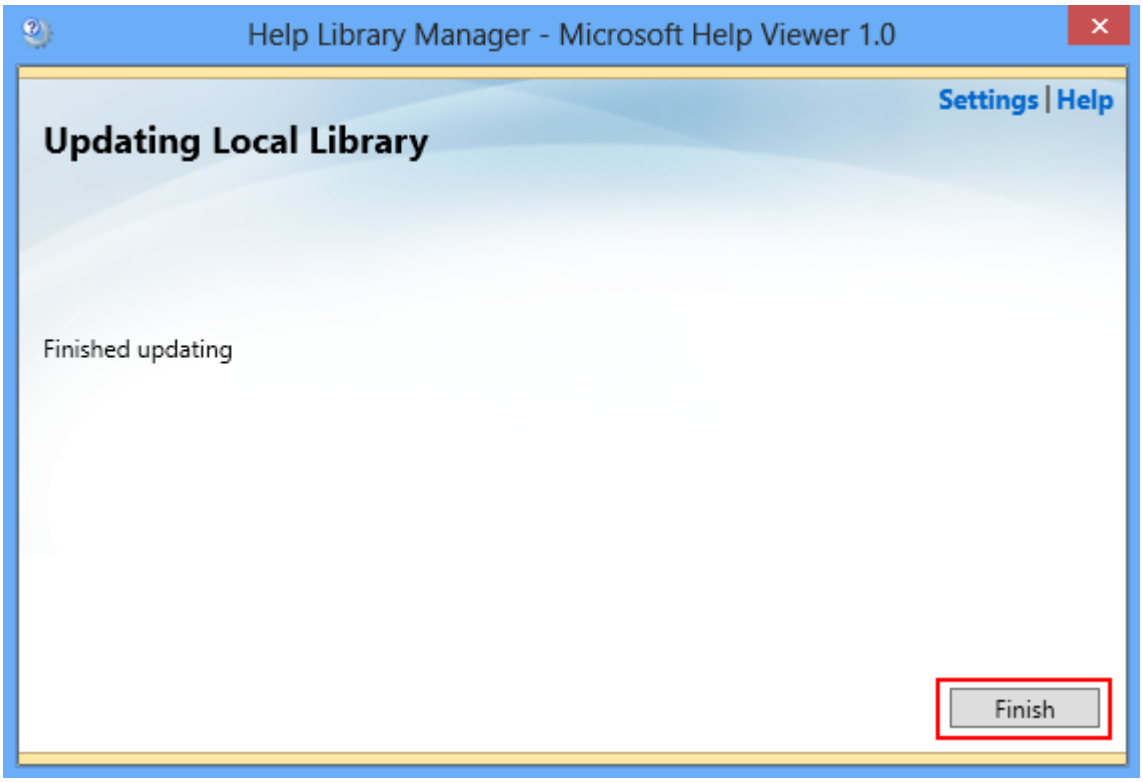
4. Click **Update** to download the available updates.



5. Confirm the installation of the update package with **Yes**.



⇨ Close the dialog after a successful update with **Finish.**



5 Reference User Interface

By default, the main commands are available in the TwinCAT user interface. To customize the menu configuration, select **Customize** from the **Tools** menu.

See also:

- Customizing the user interface [▶ 34]

5.1 File

5.1.1 Archiving options

The TwinCAT development environment offers three different archiving file types for storing a TwinCAT project, e.g. for archiving purposes or for passing it on to colleagues: tnzip, tszip and tpzip.

The type of archive file you should use depends on which projects you want to store in the archive folder.

Archiving file type	Focus	Contents	Commands
tnzip	Solution [▶ 50]	The *.tnzip archive folder contains all TwinCAT project types included in the solution. These can be TwinCAT, TwinCAT HMI, Scope or Connectivity projects.	Build [▶ 50] Open [▶ 51]
tszip	TwinCAT project [▶ 51]	The *.tszip archive folder contains the TwinCAT project to be archived.	Build [▶ 232] Open [▶ 51]
tpzip	PLC project [▶ 52]	The *.tpzip archive folder contains the PLC project to be archived.	Build [▶ 52] Open [▶ 52]

5.1.1.1 Solution

- Creating a *.tnzip TwinCAT solution archive: Command Save <solution name> as Archive... [▶ 50]
- Open *.tnzip TwinCAT solution archive: Command Open Solution from Archive [▶ 51]

5.1.1.1.1 Command Save <solution name> as Archive...

Function: The command opens the standard dialog for saving a file as an archive. The solution can be stored as a ***.tnzip** archive under the desired path.

Call: **File** menu, context menu

Requirement: The solution is selected in the **Solution Explorer**.

Content of *.tnzip	The *.tnzip archive folder contains all TwinCAT project types included in the solution. These can be TwinCAT, TwinCAT HMI, Scope or Connectivity projects.
Command for opening	A tnzip archive can be reopened with the following command: Command Open Solution from Archive [▶ 51].
Note on PLC projects	If the solution contains one or more PLC projects, the files and folders stored in the archive folder for those PLC projects will depend on the PLC project settings of the respective project. Settings tab [▶ 112]

5.1.1.1.2 Command Open Solution from Archive

Function: The command extracts a TwinCAT solution archive ***.tnzip**.

Call: Menu **File > Open**

Executing the command opens the **Open** dialog. Select the archive file from the file system and confirm the dialog. The **Select Folder for new Solution** dialog opens. Select a folder for storing the extracted solution files.

Content of *.tnzip	The *.tnzip archive folder contains all TwinCAT project types included in the solution. These can be TwinCAT, TwinCAT HMI, Scope or Connectivity projects.
Command for creating	A tnzip archive can be created with the following command: Command Save <solution name> as Archive... [▶ 50]
Note on PLC projects	If the solution contains one or more PLC projects, the files and folders stored in the archive folder for those PLC projects will depend on the PLC project settings of the respective project. Settings tab [▶ 112]

5.1.1.2 TwinCAT project

- Creating a *.tszip TwinCAT project archive: Command Save <TwinCAT project name> as Archive... [▶ 232]
- Open a *.tszip TwinCAT project archive: Command Project/Solution (Open Project/Solution) [▶ 51]

5.1.1.2.1 Command Project/Solution (Open Project/Solution)

Symbol: 

Hotkey: **[Ctrl] + [Shift] + [O]**

Function: The command opens the default dialog for opening a file. Here you can browse the file system for a TwinCAT project file and open it in the development system.

Call: Menu **File > Open**

Open Project dialog

File type	Selection list for filtering the file type • Files of all supported formats can be opened.
Options	• Add (Use this option only to add a measurement project to a solution, for example. Do not use the option to add multiple TwinCAT projects to a solution.) • Close solution
Open	TwinCAT opens the selected project file. If necessary, it is converted first.

TwinCAT *.tszip project archive

Content of *.tszip	The *.tszip archive folder contains the TwinCAT project to be archived.
Command for creating	A tszip archive can be created with the following command: Command Save <TwinCAT project name> as Archive... [▶ 232]
Note on PLC projects	If the TwinCAT project contains one or more PLC projects, the files and folders stored in the archive folder for those PLC projects will depend on the PLC project settings of the respective project. Settings tab [▶ 112]

See also:

- PLC documentation: Your first TwinCAT 3 PLC project
- PLC documentation: Creating a standard project

5.1.1.3 PLC project

- Creating a *.tpzip PLC project archive: Command Save <PLC project name> as archive... [▶ 52]
- Open *.tpzip PLC project archive: Command Add Existing Item (Project) [▶ 52]

5.1.1.3.1 Command Save <PLC project name> as archive...

Function: The command opens the standard dialog for saving a file as an archive. The PLC project can be stored as a ***.tpzip** archive under the desired path.

Call: **File** menu, context menu

Requirement: The TwinCAT PLC project (<PLC project name>) is selected in the **Solution Explorer**.

The files and folders in the archive folder depend on the PLC project settings.

Content of *.tpzip	The *.tpzip archive folder contains the PLC project to be archived.
Command for opening	A tpzip archive can be reopened with the following command: Command Add Existing Item (Project) [▶ 52]
Note on PLC projects	The files and folders stored in the archive folder for the PLC project depend on the PLC project settings of this PLC project. Settings tab [▶ 112]

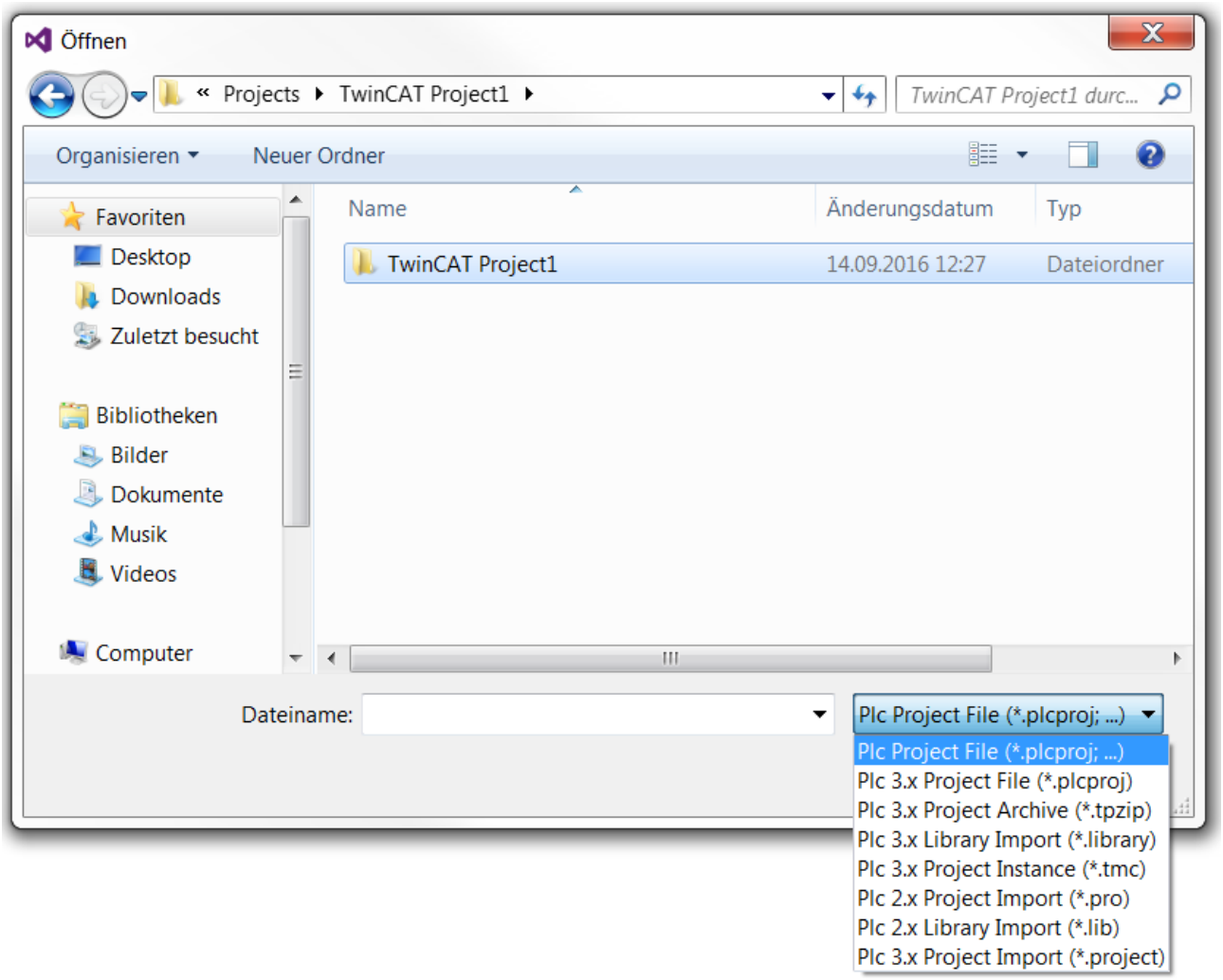
5.1.1.3.2 Command Add Existing Item (Project)

Symbol: 

Function: The command opens the standard browser dialog, which can be used to search for a PLC project file and open it in the programming system. If a suitable converter is installed, you can open projects in a different format.

Call: Project menu or PLC object context menu in the **Solution Explorer**

Requirement: The PLC node is selected in the TwinCAT project tree.



File type	By default, you can set the filter to one of the following file types: • PLC 3.x Project file (*.PLCproject): TwinCAT 3 PLC projects with the extension ".PLCproject" • PLC 3.x Project archive (*.tpzip): TwinCAT 3 PLC project archives with the extension ".tpzip" ◦ See also: Command Save <PLC project name> as archive... [▶ 52] • PLC 3.x Library import (*.library): TwinCAT 3 PLC libraries with the extension ".library" • PLC 2.x Project file (*.pro): TwinCAT 2 PLC projects with the extension ".pro" • PLC 2.x Import library (*.lib): TwinCAT 2 PLC libraries with the extension ".lib" • PLC 3.x Project import (*.PLCproject): PLC projects with the extension ".project"
Open	The selected project file is opened or converted and then opened.

***.tpzip PLC project archive**

Content of *.tpzip	The *.tpzip archive folder contains the PLC project to be archived.
Command for creating	A tpzip archive can be created with the following command: Command Save <PLC project name> as archive... [▶ 52]
Note on PLC projects	The files and folders stored in the archive folder for the PLC project depend on the PLC project settings of this PLC project. Settings tab [▶ 112]

Possible scenarios when opening a PLC project

The following scenarios are possible when you open a project:

1. Another project is still open. [▶ 54]
2. The project was saved with an older version of TwinCAT 3. [▶ 54]
3. The project was not saved with TwinCAT 3. [▶ 54]
4. The project was not terminated properly and "Save automatically" was enabled. [▶ 56]
5. The project is read-only. [▶ 56]
6. It is a library that is installed in a library repository and retrieved from it. [▶ 56]

1. Another project is still open.

You are asked if the other project should be saved and closed.

2. The project was saved with an older version of TwinCAT 3.

If the file format differs because the open project was saved with an older version of TwinCAT 3, there are two options:

- If the project cannot be saved in the format of the currently used programming system, you must update it to continue working on the project. The expression that appears at this point: **The changes you made...** refers to internal tasks of various components while the project is loaded.
- If the project can still be saved in the previous format, you can decide whether to update or retain the format. If you decide to retain the format, data loss may occur. If you decide to update the format, the project can no longer be opened with the old version of the programming system.

In addition to the file format, the versions of the explicitly inserted libraries, the visualization profile and the compiler version of the opening project may differ from those installed with the current programming system.

If newer versions are installed on the current programming system, the **Project Environment** dialog opens automatically, where you can update the versions. If no update is made at this point, this can be done later at any time via the **Options > Project Environment** dialog.

Note the compiler version

If a project is opened that was created with an older version of the programming system and for which the latest compiler version is set in the project settings, while the project environment setting for the compiler version is set to **Do not update** in the new programming system, the compiler version that was used last in the old project continues to be used (i.e. not the "Current" version in the new environment).

3. The project was not saved with TwinCAT 3.

Case 1)

If you set the file filter when selecting the project to be opened and an appropriate converter is available, the converter is used automatically and the project is brought into the current format. The conversion is converter-specific. Usually, you are prompted to define the handling of referenced libraries or device references.

TwinCAT 3 converter

Adaptation of a TwinCAT PLC control project to the TwinCAT 3 syntax can only be successful during import if the converter is able to compile the project without errors.

If you have set the **All Files** option when selecting the project to be opened, no converter is enabled and the **Convert Project** dialog opens. In the dialog, you need to explicitly trigger the conversion of the project by selecting one of the options.

- **Convert to the current format:** Select the converter you want to use from the selection list (application for conversion). After the conversion, the project can no longer be opened in the old version.
- **Create a new project and add a specific device:** (Not yet implemented)

TwinCAT 2.x PLC Control project options

The project directory path set in the TwinCAT 2.x PLC control project options and the project information are adopted in the **Project Information** dialog.

Case 2)

If libraries are integrated in the project for which "conversion mapping" has not yet been stored in the library options, the **Converting a library reference** dialog appears, in which you can define how this reference should be converted:

- **Convert and install the library:** If you select this option, the referenced library is converted to the new format and remains referenced in the project. It is automatically installed in the library repository under the **Other** category and continues to be used. If the library does not have the project information (title, version) required for an installation, you will be prompted to enter it in the **Enter Project Information** dialog.
- **Use the following library, which is already installed:** If you select the options, the referenced library is replaced by another one that is already installed on the local system. Use the **Select** button to open the **Select...** dialog. Here you can select the desired version of one of the installed libraries. This corresponds to the configuration of the version handling in the **Library Properties** dialog. An asterisk ("*") means that, as a rule, the latest version of the library available on the system is used in the project. The list of available libraries is structured in the same way as in the **Library Repository** dialog. You can sort the list by company and category.
- **Ignore the library. The reference will not appear in the converted project:** If you enable this option, the library reference is removed. The library is then no longer included in the converted project.
- **Use this mapping in future if this library is present:** If you enable this option, the settings made here in the dialog will also be applied to future project conversions, as soon as the respective library is referenced.

In the converted project, the library references are defined in the Global Library Manager in the Solution Explorer. After the conversion of the library references, the project conversion continues with the **Open Project** dialog, as described above.

For general information on library management, see section “Using libraries” in the PLC documentation.

Case 3)

When you open a TwinCAT 2.x PLC Control project that references a device (target system) for which no "conversion mapping" has yet been defined in the TwinCAT 2.x PLC Control converter options, the **Device Conversion** dialog opens, in which you can specify whether and how the old device references are to be replaced by more recent ones. The device originally used is displayed. Choose one of the following options:

- **Use the following already installed device:** Click the **Select** button to open the **Select target system** dialog, in which you can select one of the devices currently installed on the system. This device is then inserted in the **Solution Explorer** of the converted project, instead of the old one. Select the option **Select a target system...** to select one of the devices listed. The list of available devices is structured in the same way as in the **Device Repository** dialog. You can sort the list by manufacturer or by category.
- **Ignore the device. No application-specific objects will be available:** If you enable this option, no entry for the device is created in the **Solution Explorer** of the new project, i.e. the device is ignored during the conversion, and no application-specific objects, such as the task configuration, are applied.

- **Save this assignment for future reference:** If you select this option, all settings of the dialog, i.e. the displayed "conversion mapping" for the device, are saved in the TwinCAT 2.x PLC Control Converter options and applied to future conversions.

4. The project was not terminated properly and "Save automatically" was enabled.

If the **Auto Save** function was enabled in the **Load and Save** options and TwinCAT 3 PLC was not terminated regularly after the last modification of the project without saving, the **Auto Save Backup** dialog opens for handling the backup copy.

5. The project is read-only.

If the project to be opened is read-only, you are asked whether you want to open the project in write-protected mode or whether you want to unlock it.

6. It is a library that is installed in a library repository and retrieved from it.

An error message is displayed if you try and open a library project that is installed in a library repository. You cannot open a library project using this path. After closing the dialog with **OK**, the project name appears in the title bar of the user interface. An asterisk ("*") after the name indicates that the project has been modified since it was last saved.

See also:

- PLC documentation: Open a TwinCAT 3 PLC project
- PLC documentation: Open a TwinCAT 2 PLC project

5.1.2 Command Project... (Create new TwinCAT project)

Symbol: 

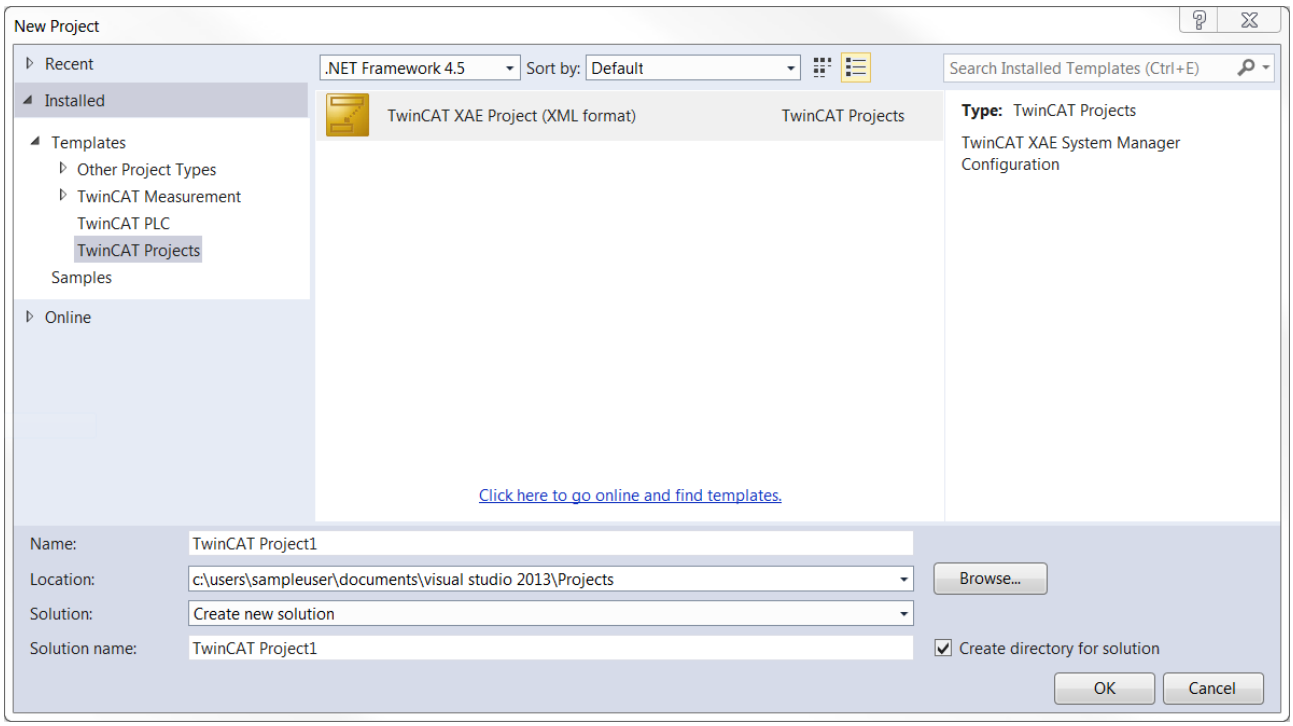
Hotkey: **[Ctrl] + [Shift] + [N]**

Function: The command opens the **New Project** dialog for creating a new TwinCAT project file.

Call: Menu **File > New**

New Project dialog

Depending on the selected template, you will obtain a project that automatically has a certain number of objects.



Categories

Recent	Shows the last used project template.
Installed > Templates	Shows the TwinCAT project templates: • Other Project Types • TwinCAT Measurement • TwinCAT PLC • TwinCAT Projects
Online	Not relevant

Templates

Category Other Project Types	
Visual Studio Solutions	Empty Visual Studio Solution
Category TwinCAT Measurement	
BodePlot	Bode Plot
Scope	Scope YT Project Scope YT NC Project Scope YT Project with Reporting Scope XY Project Scope XY Project with Reporting
Category TwinCAT PLC	
	TwinCAT PLC Project
Category TwinCAT Projects	
	TwinCAT XAE Project

Name	Name of the project to be created. A default name appears, depending on the template. The numeric supplement ensures the uniqueness of the name in the file system. You can change the file name according to the file path conventions of the operating system. Dots are not allowed in the name. TwinCAT automatically adds the file extension that matches the selected template.
Location	Location for the new project file. The **Browse**... button opens a dialog for browsing the file system. The combobox field shows the history of previously entered paths.
Solution	• Create new solution • Add • Create in new instance
Create directory for solution	☑ Solution directory is created.
Solution name	Name of the solution. By default, the TwinCAT project name is automatically adopted.
OK	TwinCAT opens a new project.

See also:

- PLC documentation: Your first TwinCAT 3 PLC project
- PLC documentation: Creating a standard project

5.1.3 Command Project/Solution (Open Project/Solution)

Symbol: 

Hotkey: **[Ctrl] + [Shift] + [O]**

Function: The command opens the default dialog for opening a file. Here you can browse the file system for a TwinCAT project file and open it in the development system.

Call: Menu **File > Open**

Open Project dialog

File type	Selection list for filtering the file type • Files of all supported formats can be opened.
Options	• Add (Use this option only to add a measurement project to a solution, for example. Do not use the option to add multiple TwinCAT projects to a solution.) • Close solution
Open	TwinCAT opens the selected project file. If necessary, it is converted first.

TwinCAT *.tszip project archive

Content of *.tszip	The *.tszip archive folder contains the TwinCAT project to be archived.
Command for creating	A tszip archive can be created with the following command: Command Save <TwinCAT project name> as Archive... [▶ 232]
Note on PLC projects	If the TwinCAT project contains one or more PLC projects, the files and folders stored in the archive folder for those PLC projects will depend on the PLC project settings of the respective project. Settings tab [▶ 112]

See also:

- PLC documentation: Your first TwinCAT 3 PLC project
- PLC documentation: Creating a standard project

5.1.4 Command Open Project from Target

Function: The command loads a project from the target system.

Call: Menu **File > Open**

Requirement: The network path to the target system must be configured.

Executing the command opens an overview of all devices on the network. Select the target system from this overview. The **Select Folder for new Solution** dialog opens.

5.1.5 Command New Project... (Add new TwinCAT project)

Function: The command opens the **New Project** dialog for creating a further TwinCAT project file in the solution.

Call: Menu **File > Add**

Requirement: A TwinCAT project is opened.

Use this command only to add a measurement project to a solution, for example. Do not use this command to add multiple TwinCAT projects to a solution. This function is currently not supported by TwinCAT.

5.1.6 Command Existing Item... (Add existing TwinCAT project)

Function: The command opens the **Add Existing Item** dialog for adding a TwinCAT project file to the solution.

Call: Menu **File > Add**

Requirement: A TwinCAT project is opened.

Use this command only to add a measurement project to a solution, for example. Do not use this command to add multiple TwinCAT projects to a solution. This function is currently not supported by TwinCAT.

5.1.7 Command Recent Projects and Solutions

Function: The command opens the list of recently used projects, from which you can select a project to open.

Call: Menu **File**

See also:

- PLC documentation: Creating and configuring a project

5.1.8 Command Save All

Symbol: 

Function: The command saves all objects of the TwinCAT project.

Call: File menu, standard toolbar options

5.1.9 Command Save

Symbol: 

Hotkey: **[Ctrl] + [S]**

Function: The command saves the solution, the TwinCAT project, the TwinCAT PLC project or a selected PLC object (Main, GVL, ...) under the current name.

Call: File menu, standard toolbar options

Requirement: The solution, the TwinCAT project object, the PLC project object (<PLC project name> Project) or the PLC object to be saved is selected in the **Solution Explorer**.

Save object

The command saves the object under the current name. If the object has been changed since the last save, the "disk" icon for the object is red, and the name in the title bar of the editor for the open object is marked with an asterisk ("*").

See also:

- Command Save <TwinCAT project name> as [▶ 60]

5.1.10 Command Save <Solution name> as

Function: The command opens the default dialog for saving a file. The solution can be stored under the desired path. By default, the file type UTF-8 solution file (* .sln) is selected.

Call: Menu **File**

Requirement: The solution is selected in the **Solution Explorer**.

5.1.11 Command Save <TwinCAT project name> as

Function: The command opens the default dialog for saving a file. The project can be stored under the desired path and file type. By default, the file type TwinCAT XAE project (*.tsproj) is selected.

Call: Menu **File**

Requirement: The TwinCAT project object is selected in the **Solution Explorer**.

Note that during this saving operation only the *.tsproj file, for example, is created in a different location. The referenced projects and the objects contained in them are not stored at this new location (for example, a PLC project that is integrated in the TwinCAT 3 project and its objects).

See also:

- Command Save [▶ 60]
- PLC documentation: Library creation

5.1.12 Command Save <PLC project name> as

Function: The command opens a dialog in which a destination directory for the PLC project file can be specified. The PLC project objects and the .plcproj file are stored in the selected directory.

Call: PLC project object context menu

Requirement: The PLC project object (<PLC project name>) is selected in the **Solution Explorer**.

5.1.13 Command Send by E-Mail...

Symbol: 

Function: The command starts the email program that is set in the system and opens a new email with the archive file of the selected project as an attachment.

Call: File menu, context menu

5.1.14 Command Close Solution

Symbol: 

Function: The command closes the currently open project. TwinCAT remains open.

Call: File menu or implicitly when opening a new/different project while a project is still open.

If the project contains unsaved changes, a query appears asking if the project should be saved.

If you have not yet explicitly saved the project, a query is displayed to confirm whether you want to delete the project files.

See also:

- PLC documentation: Creating and configuring a PLC project

5.1.15 Command Close

Function: The command closes the open editor.

Call: Menu **File**

Requirement: The editor to be closed is active, or the object is selected in the PLC project tree.

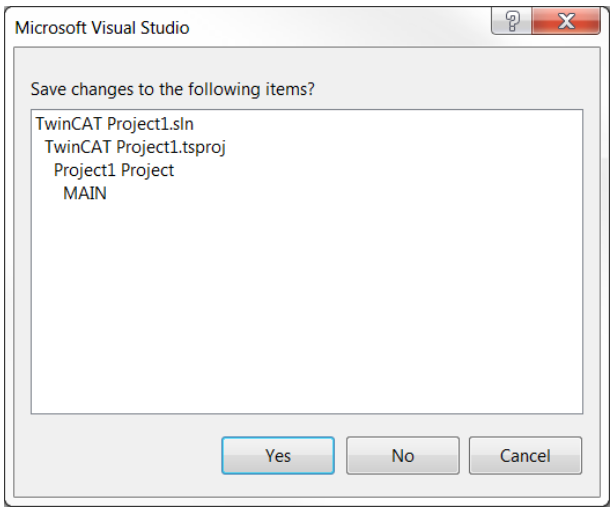
5.1.16 Command Exit

Symbol: 

Hotkey: **[Alt] + [F4]**

Function: The command stops the programming system. If a project that has been modified since the last time it was saved is currently open, a dialog appears asking whether you want to save the project.

Call: Menu **File**



5.1.17 Command Page Setup…

Symbol: 

Function: The command opens the **Page settings** dialog for configuring the layout for the print version of the project content.

Call: Menu **File**

Requirement: An editor window is active.

See also:

- Command Print [▶ 62]

5.1.18 Command Print

Symbol: 

Function: The command opens the standard Windows dialog for printing documents.

Call: Menu **File**

Requirement: An editor window is active.

5.2 Edit

5.2.1 Standard Commands

TwinCAT offers the following standard commands:

- Undo

   , hotkeys: **[Ctrl] + [Z]**
- Redo:

   , hotkeys: **[Ctrl] + [Y]**
- Cut

 , hotkeys: **[Ctrl] + [X]**

- Copy:

 , hotkeys: **[Ctrl] + [C]**

- Paste:

 , hotkeys: **[Ctrl] + [V]**

- Delete:

 , hotkeys: **[Del]**

- Select All: Hotkey: **[Ctrl] + [A]**

Call: Edit menu, PLC project tree context menu, Editor window context menu

Not all editors support the **Paste** command, or its application may be limited. In graphical editors, the command is only supported if the paste operation creates a correct construct.

In the PLC project tree, the command refers to the currently selected object. Multiple selections are possible.

5.2.2 Command Delete

Hotkey: **[Del]**

Function: The command removes the selected PLC object from the solution. The object is retained in the project directory.

Call: PLC object context menu

5.2.3 Command Select All

Hotkey: **[Ctrl + A]**

Function: The command selects the entire content.

Call: Edit menu, Editor window context menu

5.2.4 Command Input Assistant

Symbol: 

Hotkey: **[F2]**

Function: The command opens the **Input Assistant** dialog, which provides support for selecting a programming element, depending on the current cursor position.

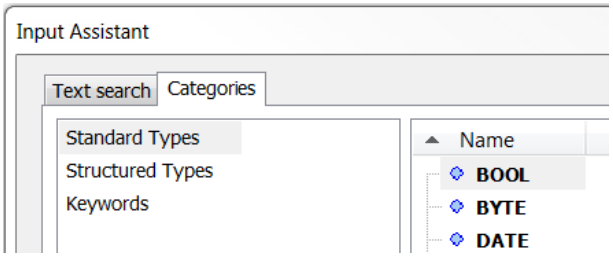
Call: Edit menu, Editor window context menu

Requirement: A POU is open in the editor, and the cursor is in a program line.

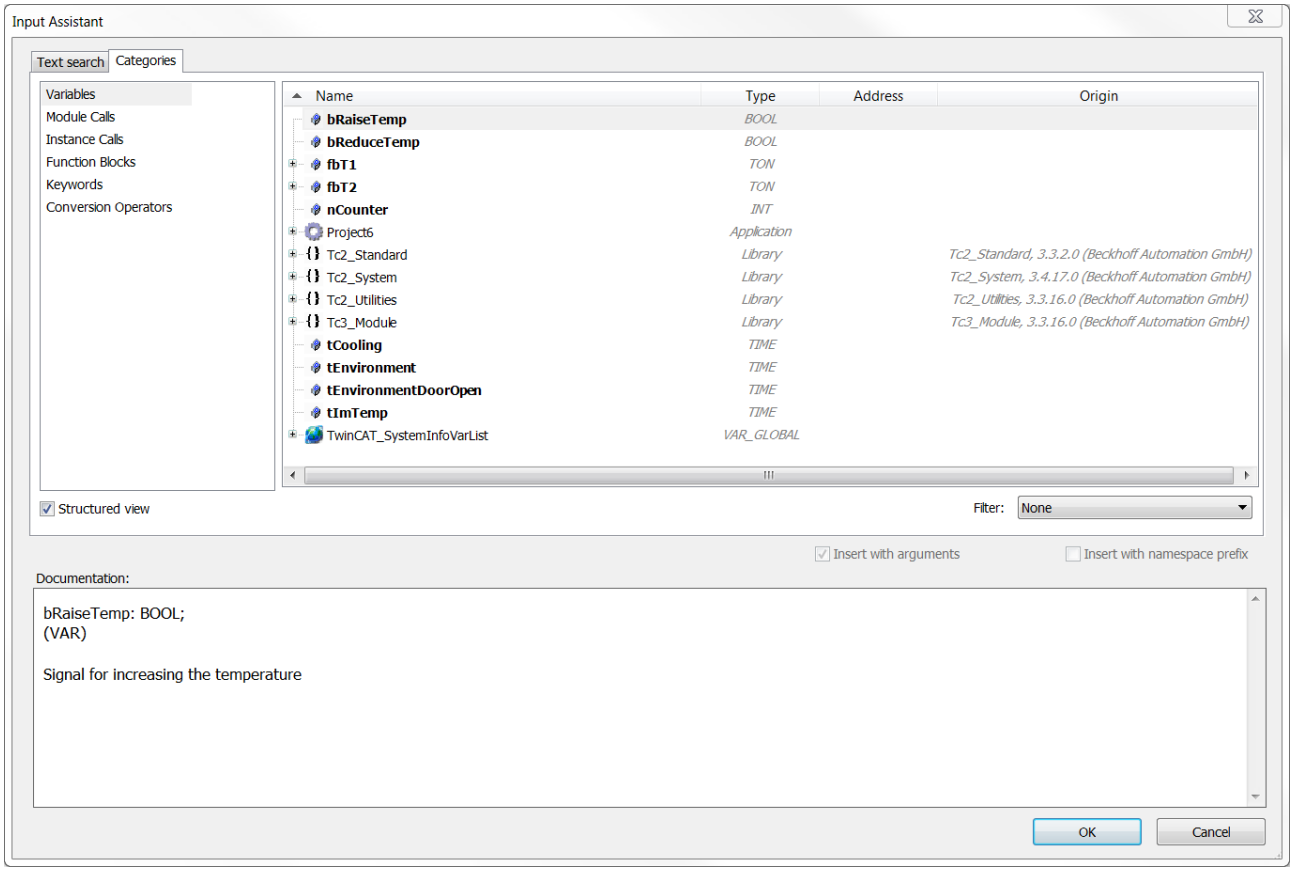
Input Assistant dialog - Categories tab

The dialog offers all programming elements, which you can add in the editor at the current cursor position. The elements are sorted by categories. For the **Variables** category you can set an additional filter for the variable type, such as local variables, global variables, constants etc.

Detail of the **Input Assistant** dialog in the declaration part of the editor:



Input Assistant dialog in the implementation part of the editor:



Structured view	☑: The dialog shows the elements in a tree structure. You can hide or show the column Type, Address and Origin by right-clicking on the column title in a submenu. ☐: The elements are shown in a flat structure.
Filter	You can set an additional filter for the variable type in the drop-down list box.
Documentation	Shows a description of the selected element.
Insert with arguments	☑: TwinCAT inserts elements that have arguments, such as functions, at the cursor position with these arguments. Example: If you insert the function block fb1, which contains an input variable fb1_in and an output variable fb1_out, "with arguments", it looks as follows in the editor: fb1(fb1_in:= , fb1_out=>)
Insert with namespace prefix	☑: TwinCAT inserts the selected element with the namespace prefixed. In the case of library function blocks, you cannot use the check box if the properties of the library specify that the namespace is mandatory.

Input Assistant dialog - Text search tab

You can use the tab to search for specific objects. When you type one or more characters into the search box, the results window lists the names of all objects whose name contains this search string. Double-click the desired object to insert it in the editor at the current cursor position.

Filter	Restrict the search to a specific variable category.

See also:

- PLC documentation: Using the input wizard

5.2.5 Command Auto Declare

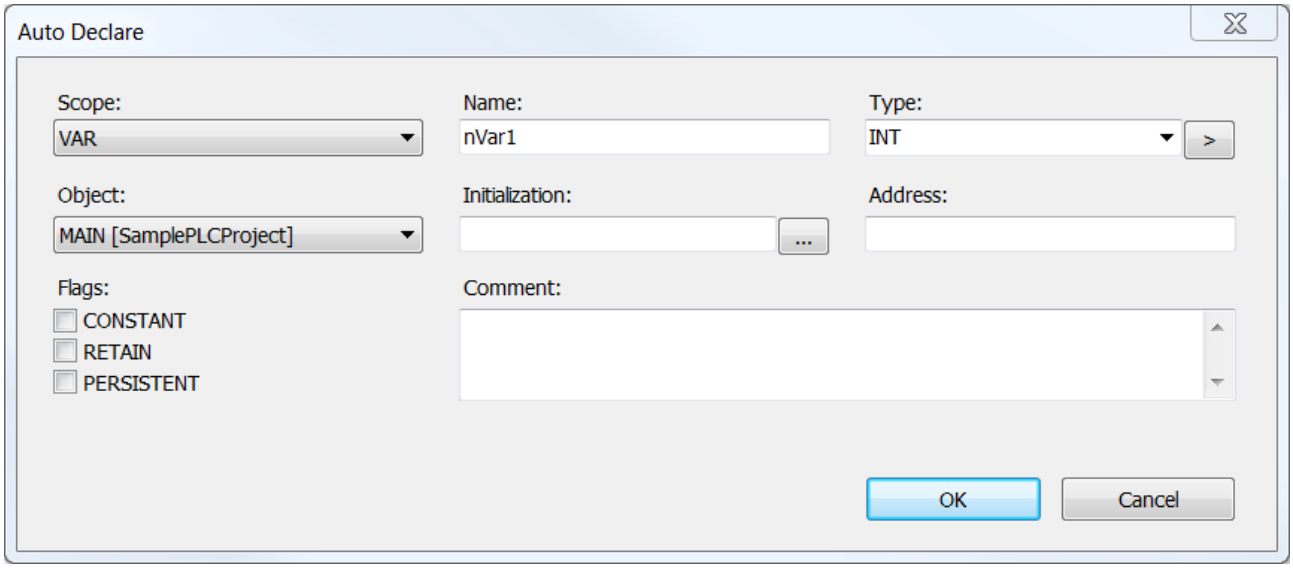
Function: The command opens the **Auto Declare** dialog, which supports the declaration of a variable.

Call: Edit menu, Editor window context menu

Requirement: A POU is open in the editor, and the cursor is in a program line.

The auto-declaration function opens the **Auto Declare** dialog even if the cursor is in the implementation part of a POU in a line that contains the name of an undeclared variable. To do this, you must have activated the option **Declare unknown variables automatically (AutoDeclare)** in the TwinCAT options (**Tools > Options > TwinCAT > PLC Environment > Smart coding**).

Auto Declare dialog



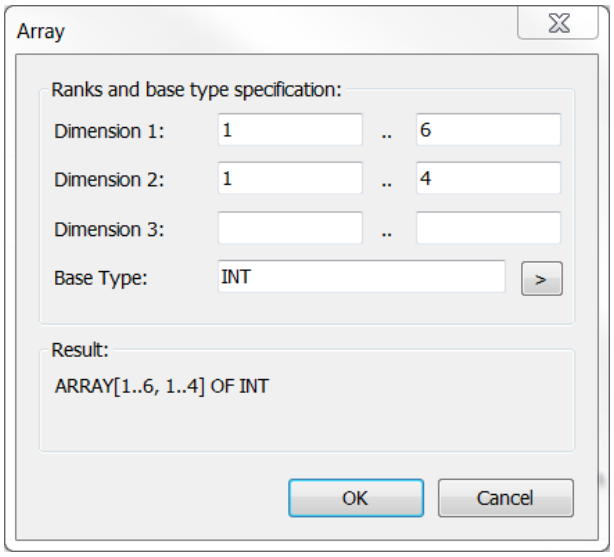
Scope	Validity range of the variables that have not yet been declared. Example: `VAR` (default setting for local variable)
Name	Variable name not yet declared Example: `bVar`
Data type	: Lists the standard data types. : • Input Assistant: Opens the **Input Assistant** dialog • Array assistant: Opens the **Array** dialog Example: `BOOL`
Object	Object in which the new variable is declared. By default, this is the object you are editing. : Lists the objects in which the variable can be declared. If no objects are available for the selected scope, the entry <Create object> appears. If you select the entry <Create object>, the dialog **Add object** opens, which can be used to create a suitable object.
Initialization value	If you do not enter an initialization value, the variable is initialized automatically. : Opens the **Initialization** dialog. This approach is helpful for initializing structured variables. Example: `FALSE`
Address	Memory address (see PLC documentation: Addresses) Example: %IX1.0
Flags	Attribute keywords • CONSTANT: Keyword for a constant • RETAIN: Keyword for a remanent variable • PERSISTENT: Keyword for a persistent variable (stricter than RETAIN) The selected attribute keyword is added to the variable declaration.
Comment	The tabular declaration editor displays the comment that was entered in the Comment column. In the textual declaration editor, it is displayed above the variable declaration. Example: `New variable`
Apply changes using refactoring	This option appears for the following validity ranges: • Input variable (VAR_INPUT) • Output variable (VAR_OUTPUT) • Input and output variable (VAR_IN_OUT) This option is automatically enabled if the autodeclaration options **On renaming variables** and **On adding or removing variables, or on changing the scope** are enabled in the TwinCAT options (**Tools > Options > TwinCAT > PLC Environment > Refactoring**) (see Dialog Options - Refactoring [▶ 159]). If this option is enabled, the variable is not yet declared when the dialog is closed. Instead, the **Refactoring** dialog opens, in which you can edit the changes further.

OK	The variable is declared and appears in the declaration.
	Example:
	`VAR`
	`    // New variable`
	`    bVar: BOOL := FALSE;`
	`END_VAR`

See also:

- PLC documentation: Declaring variables

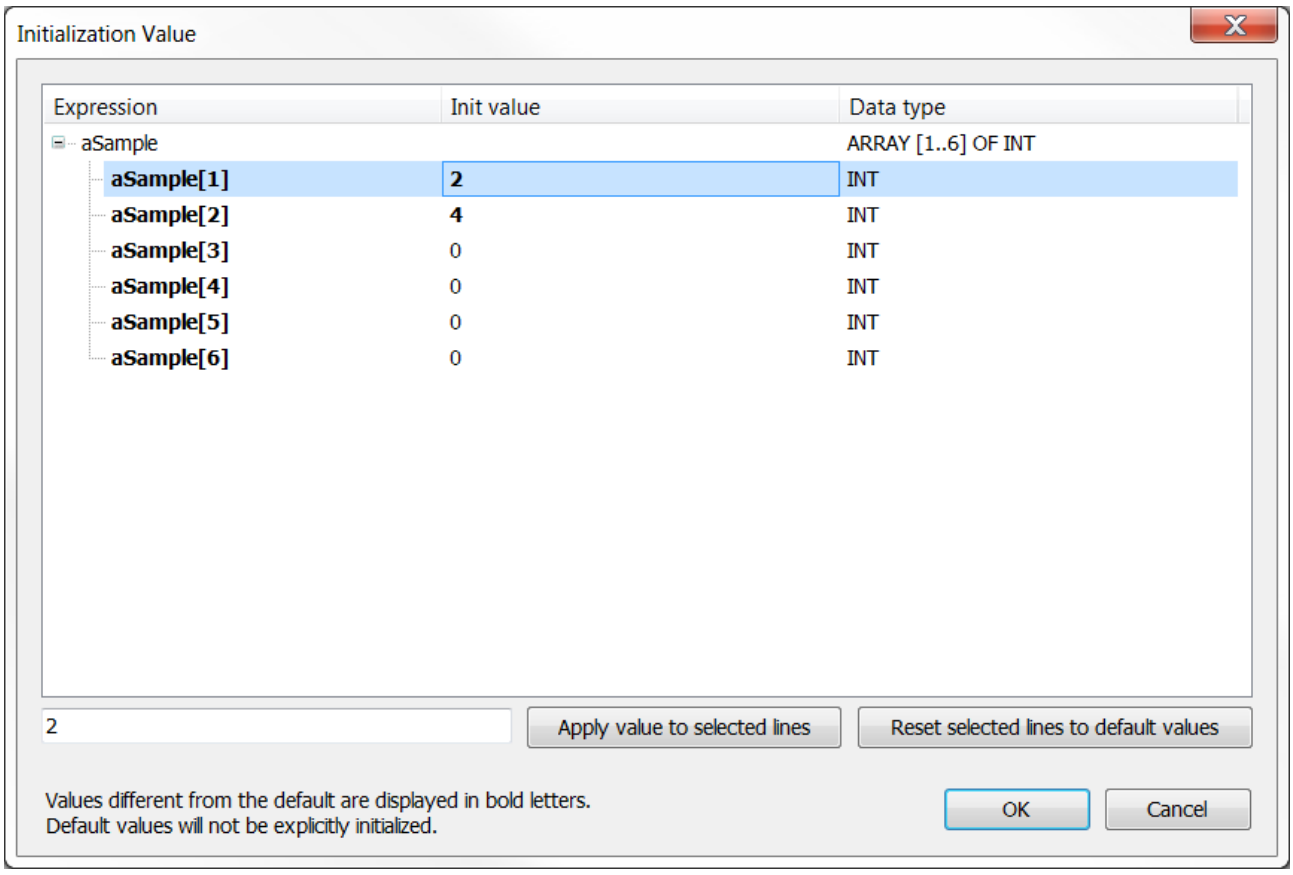
Array dialog



Ranks and base type specification	Definition of the field sizes (dimension) by entering the lower and upper bounds and the base type of the array. You can enter the base type directly or by using the **Input Assistant** or **Array** dialog, if you click the > button.
Result	Display of the defined array.

TwinCAT only reinitializes variables if you have changed the initialization values of the variables.

Initialization Value dialog



Listing of the variables with name (expression), initialization value and data type. Changed initialization values are shown in bold.	
Input field below the list	Enter an initialization value for the selected variable(s).
Apply value to selected lines	Change the initialization value of the selected line(s) according to the value of the input field.
Reset selected lines to default values	Setting the default initialization values.
OK	TwinCAT adopts the initialization values in the **Auto Declare** dialog.

See also:

- PLC documentation: Using the input wizard

5.2.6 Command Add to Watch

Symbol: 

Function: The command adds the variable on which the cursor is currently positioned to a Watch List for online monitoring.

Call: Context menu

Requirement: The PLC project is in online mode and the cursor is placed on a variable in an editor.

The command inserts the variable into the currently open Watch List. If no Watch List is currently open, the command inserts the variable into Watch List 1 and opens its view.

See also:

- PLC documentation: Using Watch Lists
- PLC documentation: Monitoring Values

5.2.7 Command Browse Call Tree

Symbol: 

Function: The command opens the **Call Tree** view, which displays the calls of the function block and its callers.

Call: Editor window context menu

Requirement: A function block is open in the editor, and the cursor is placed in a variable.

5.2.8 Command Go To

Function: Command moves the cursor to a defined line of code.

Call: Menu **Edit**

Requirement: A text editor is open, and the cursor is in a program line.

The command opens a dialog with a **Line number** input field.

5.2.9 Command Go To Definition

Symbol: 

Hotkey: **[F12]**

Function: The command shows the definition point of a variable or function.

PLC editor

Call: Editor window context menu

Requirement: A POU is open in the editor, and the cursor is on a variable or function.

PLC process image

Call: Solution Explorer context menu

Requirement: The process image (project instance) is expanded and the cursor is positioned at an allocated variable in the process image.

5.2.10 Command Go to Instance

Symbol: 

Function: The command opens the instance of a function block in a new window.

Call: Editor window context menu

Requirement: The PLC project is in online mode. A POU is open in the editor, and the cursor is positioned on the instance of a function block.

The command is not available for temporary instances or instances from compiled libraries.

5.2.11 Command Find all references

Hotkey:**[Shift+F12]**

Function: The command displays all locations where a variable is used in the **Cross Reference List** view.

Call: Context menu

Requirement: A POU is open in the editor and the cursor is in a variable, or the **Cross Reference List** view is open and a variable is specified in the **Name** field.

See also:

- PLC documentation: Find locations where the cross reference list is used

5.2.12 Command Navigate To

Function: The command opens a dialog for selecting a specific element to be opened.

Call: Menu **Edit**

5.2.13 Command Make Uppercase

Function: The command converts all lower-case letters in the selected code to upper-case letters.

Call: Menu **Edit > Advanced**

Requirement: A POU is open in the editor, and code is selected.

5.2.14 Command Make Lowercase

Function: The command converts all upper-case letters in the selected code to lower-case letters.

Call: Menu **Edit > Advanced**

Requirement: A POU is open in the editor, and code is selected.

5.2.15 Command View white spaces

Symbol: 

Function: The command causes control characters to be displayed for spaces and tabs.

Call: Menu **Edit > Advanced**

Requirement: A POU is open in the editor.

TwinCAT visualizes spaces by a dot and tabulators by an arrow.

5.2.16 Command Comment Selection

Hotkey: [Ctrl+K] + [Ctrl+C]

Function: The command comments out a selected code section. The code section is excluded from the compilation and has no influence on the program execution.

Call: Menu **Edit > Advanced**

Requirement: A function block is open in the editor, and code is selected.

This command can be used to create comments for the documentation of a program or program section, or to temporarily exclude a code section from compilation. The command Command Uncomment Selection [▶ 70] can be used to cancel a comment, so that a commented-out code section is reintegrated into the program execution.

5.2.17 Command Uncomment Selection

Hotkey: [Ctrl+K] + [Ctrl+U]

Function: The command cancels a comment and reintegrates a commented-out code section back into the program execution.

Call: Menu **Edit > Advanced**

Requirement: A function block is open in the editor and code is selected which was previously commented out using the command Command Comment Selection [▶ 70].

5.2.18 Command Quick Find

Symbol: 

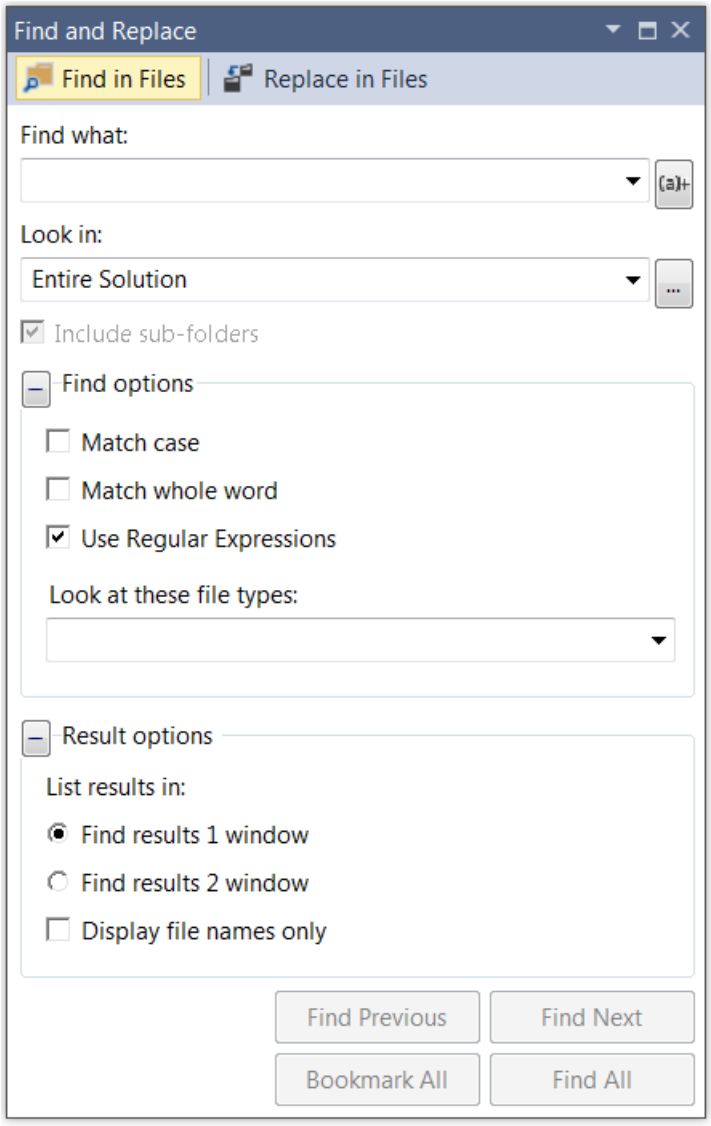
Hotkey: **[Ctrl] + [F]**

Function: The command scans the project or parts of it for a specific string.

Call: Menu **Edit > Find and Replace**

The command opens the **Find and Replace** dialog (**Find in Files** button is active), in which you enter the search string and the search options.

Find and Replace dialog



Replace in Files	Switches to the **Find and Replace** dialog (button **Replace in Files** is active)
Find what	Search string.
Look in	 : Selection list with the objects that are searched: Entire solution: All editable places in all objects of the project are scanned. Current project: All open documents: All editors that are currently open in a window are scanned. Current document: Only the editor in which the cursor is currently located is searched.  : Opens a dialog in which the objects to be searched can be defined more precisely.
Match case	 : The search is case-sensitive.
Match whole word	 : Only strings that exactly match the search string are found.
Look at these file types	Drop-down list for selecting a file type
Use Regular Expressions	Enables the  button, which will help you enter regular expressions. This function ist not supported for PLC editors!
Display file names only	 : Only file names are displayed.
Find Next	Start the search. The next search result is displayed at its position in the corresponding editor.
Find All	All search results are shown in the message window. The object and the exact position of the search result are displayed. • (Decl): Declaration part of the object • (Impl): Implementation part of the object Double-click on the list entry to display the search result in the editor.

See also:

- Command Quick Replace [▶ 72]
- PLC documentation: Find and Replace in the entire project

5.2.19 Command Quick Replace

Symbol: 

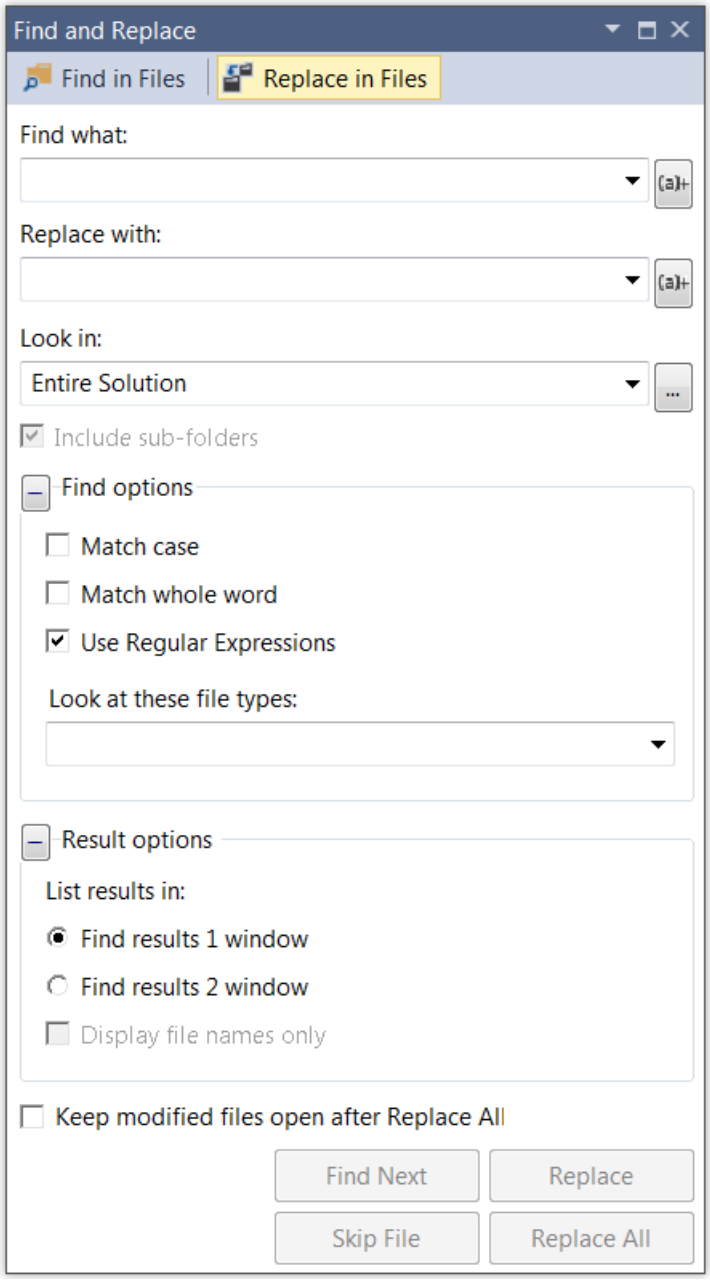
Hotkey: **[Ctrl] + [H]**

Function: The command scans the project or parts of it for a specific string and replaces it.

Call: Menu **Edit > Find and Replace**

The command opens the **Find and Replace** dialog (**Replace in Files** button is active), in which you enter the string to be replaced and the new string, along with the search options.

Find and Replace dialog



In addition to the options in the "Find" dialog, the following settings are possible:

Replace with	Input field for the new string.
Replace	The next string that is found is highlighted in the editor and replaced (step-by-step replacement).
Replace All	All strings that are found are replaced at once, without being displayed in the editors.
Keep modified files open after Replace All	The editors of the found objects remain open.

See also:

- Command Quick Find [▶ 70]
- PLC documentation: Find and Replace in the entire project

5.2.20 Command Switch write mode

Hotkey: **[Insert]**

Function: This command activates overwrite mode or insert mode.

Call: Double-click **[INS]** or **[OVR]** in the status and information bar

Requirement: An editor window is active.

If overwrite mode is active, characters following the cursor are overwritten when new characters are entered. If insert mode is active, characters are inserted, and existing characters following the cursor are retained.

5.2.21 Command Rename

Function: The command allows you to rename a PLC object in **Solution Explorer**.

Call: PLC object context menu

5.2.22 Command Edit object (offline)

Function: The command opens the object offline in its editor.

Call: Project menu, context menu

Requirement: The PLC project is in online mode. An object is selected in the PLC project tree.

This means that you can also edit the object in online mode. The change is then transferred to the controller with the command **Online Change** or **Download**.

See also:

- Command Online Change [▶ 138]
- Command Download [▶ 137]

5.2.23 Command Rename '<variable>'

Function: The command opens the **Rename** dialog for renaming an object or a variable.

Call: Context menu PLC object, context menu Editor window > Refactoring

Requirement: An object is selected in the PLC project tree, or the cursor is positioned before or on a variable identifier in the declaration part of a programming object.

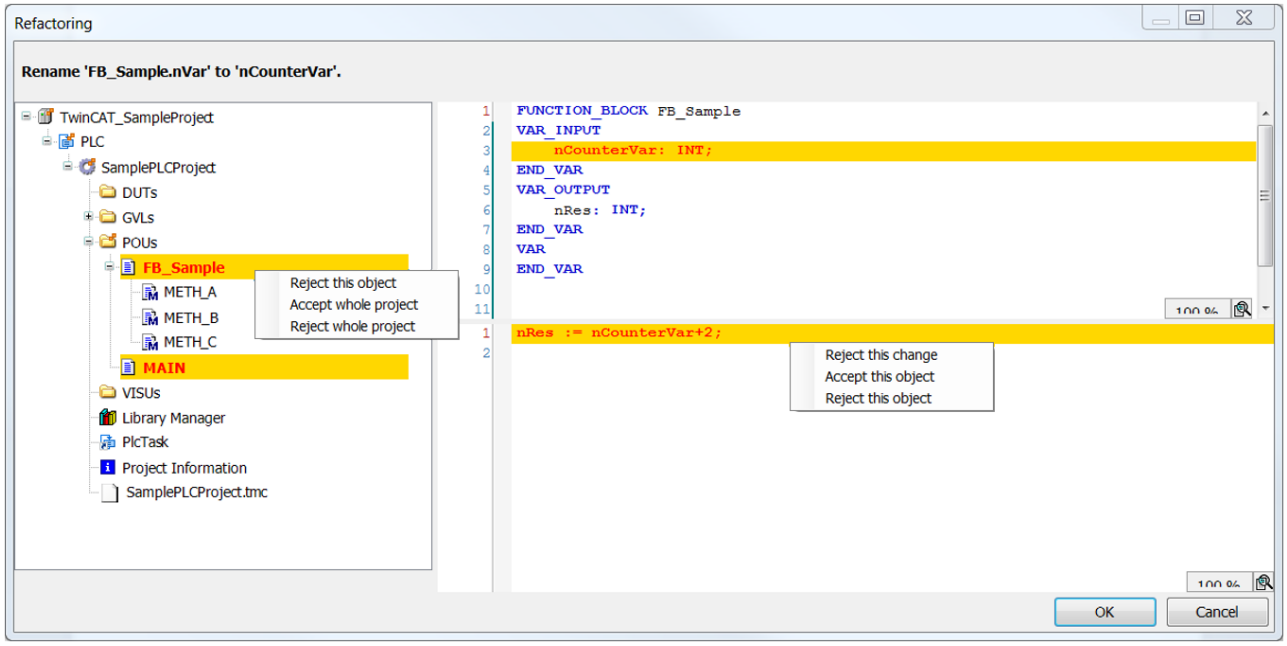
You can rename:

- Variables
- POUs
- GVLs
- Methods
- Properties

Rename dialog

Current name	Name of the object or variable
New name	Input field for a new name. If the name that is entered already exists, TwinCAT reports this directly under this input field.
OK	Can be activated if you have entered a valid name in **New Name**. Opens the **Refactoring** dialog. The respective objects and locations are color-coded in both windows. In both windows, you can specify an action for each location. Various commands are available in the context menu.

Refactoring dialog



The dialog shows all the usage locations within the project. The respective objects and locations are highlighted in color.	
Left dialog part	Navigation tree of the project with the respective object.
Right part of the dialog	Displays the position within an object where the current name occurs.
In both windows, you can specify an action for each location. The commands described below are available in the context menu.	
Reject this change	Discarding the individual change in the right part of the dialog.
Accept this object	Accept all changes in the affected object
Reject this object	Discard all changes in the affected object
Accept the entire project	Accept all changes in the project
Reject the entire project	Discard all changes in the project
TwinCAT shows accepted changes with a yellow background, discarded changes with a gray background.	

See also:

- PLC documentation: Refactoring

5.2.24 Command Add '<variable>'

Symbol: 

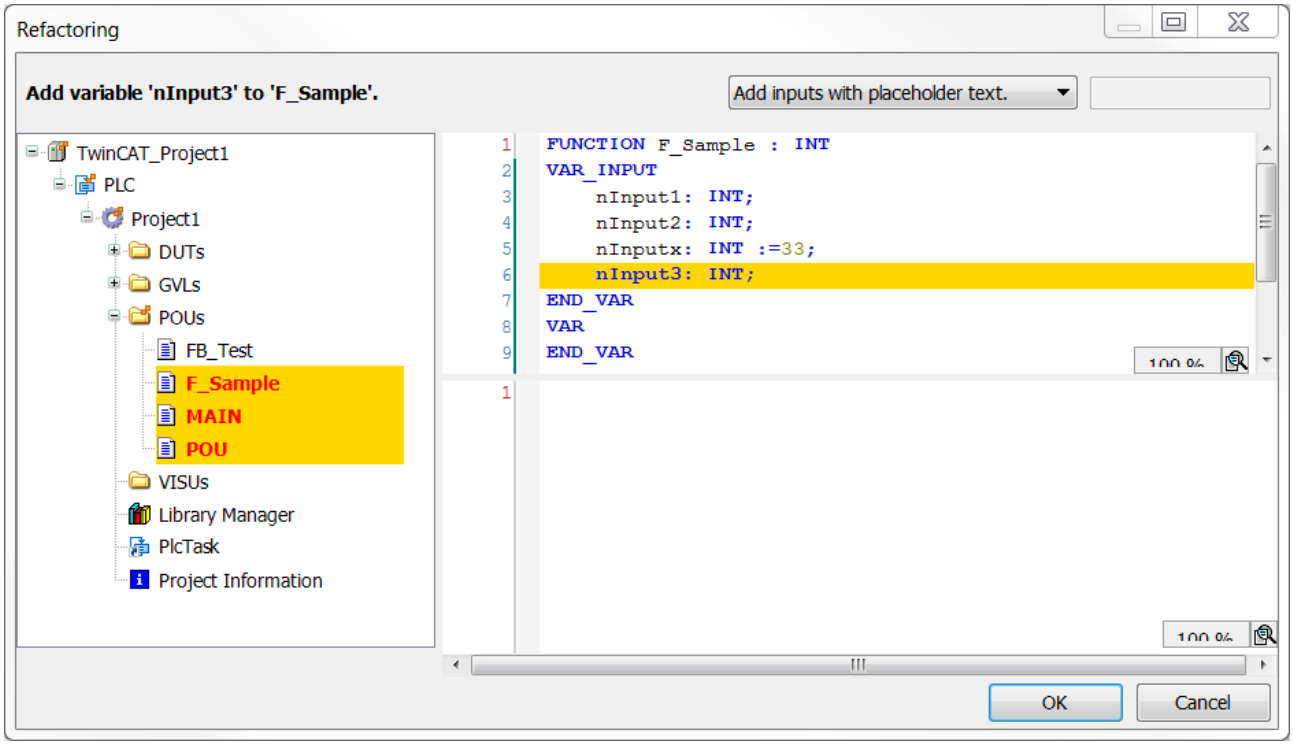
Function: This command allows you to declare a new variable in a POU and automatically update it at the point of use of the POU.

Call: Context menu Editor window > Refactoring

Requirements: The focus is on the declaration part of a POU.

The command opens the default dialog for declaring the variable. After closing the declaration dialog with **OK** the two-part **Refactoring** dialog appears.

Refactoring dialog



Left dialog part	Navigation tree of the project. Color-coding of the function blocks in which the POU is used: Red font with yellow background. After clicking on the POU object, the detailed view opens in the right part of the dialog.
Right part of the dialog	Declaration part and implementation of the POU in whose declaration the variable is added. Color-coding of the change points: Newly added declaration in blue font and yellow background.

Before you decide which changes you want to apply at which points, select the desired option from the drop-down list at the top right:

Add inputs with placeholder text	Default placeholder text: _REFACTOR_; editable The placeholder text defined here appears at the usage points of the newly added variables in the implementation code. It is used to find the affected locations.
Add inputs with the following value	Initialization value for the new variable

Commands for accepting or rejecting the change(s) are available in the context menu of the change point(s), both in the left and right part of the dialog. See also the description of the Command Rename '<variable>' [▶ 74].

Examples:

1. The function F_Sample is assigned a new input variable “nInput3” with initialization value “1” via refactoring. The change has the following effect:

Before:

```
F_Sample(nVarA + nVarB, 3, TRUE);
F_Sample(nInput1:= nVarA + nVarB, nInput2 :=3 , nInputx := TRUE);
```

After:

```
F_Sample(nVarA + nVarB, 3, TRUE, nInput3 := 1);
F_Sample(nInput1:= nVarA + nVarB, nInput2 :=3 , nInputx := TRUE, nInput3 := 1);
```

2. The function F_Sample is assigned a new input variable “nInput3” with placeholder text “_REFACTOR_” via refactoring:

Before:

```
F_Sample(nInput1 := nVarA + nVarB, nInput2 := 3, nInputx := TRUE);
F_Sample(nVarA + nVarB, 3, TRUE);
```

After:

```
F_Sample(nInput1 := nVarA + nVarB, nInput2 := 3, nInputx := TRUE, nInput3 := _REFACTOR_);
F_Sample(nVarA + nVarB, 3, TRUE, nInput3 := _REFACTOR_);
```

See also:

- PLC documentation: Refactoring
- PLC documentation: Auto Declare dialog [▶ 65]

5.2.25 Command Remove '<variable>'

Symbol: 

Function: The command removes an input or output variable from the POU and all POU usage points.

Call: Context menu Editor window > Refactoring

Requirements: The cursor is on the identifier of the variable to be removed in the declaration part of the POU.

The command first opens a dialog showing the information about the desired removal. After confirmation, the **Refactoring** dialog appears. A description of the dialog can be found in section “Command Add '<variable>' [▶ 75]”.

If you accept the changes in **Refactoring** dialog, the corresponding input or output parameters are deleted at the usage points of the affected POU.

In CFC, only the connection of the removed input or output to the function block is removed. The input or output itself is retained in the chart.

Sample in ST:

You use **Refactoring** to remove the input variable “nInput4” in a POU. An automatic adjustment is made at the respective usage points:

Before the removal:

```
F_Sample(nInput1 := nVarA + nVarB, nInput2 := 3, nInput4 := 1, nInput5 := TRUE);
F_Sample(nVarA + nVarB, 3, 1, TRUE);
```

After the removal:

```
F_Sample(nInput1 := nVarA + nVarB, nInput2 := 3, nInput5 := TRUE);
F_Sample(nVarA + nVarB, 3, TRUE);
```

See also:

- PLC documentation: Refactoring

5.2.26 Command Reorder variables

Symbol: 

Function: In the declaration editor, this command enables you to change the sequence of the variables in the currently focused scope VAR_INPUT, VAR_OUTPUT or VAR_IN_OUT.

Call: Context menu of the currently focused scope in the declaration editor

Requirement: The focus is in the declaration of one of the above scopes, and more than one variable is declared in it.

The command opens the **Rearrange** dialog with a list of all declarations of the currently focused scope. You can use the mouse to drag a selected declaration up or down to another position.

See also:

- PLC documentation: Rearranging variables in the declaration

5.3 View

5.3.1 Command Open object

Symbol: 

Function: The command opens the object in its editor.

Call: View menu, context menu PLC object, double-click the PLC object

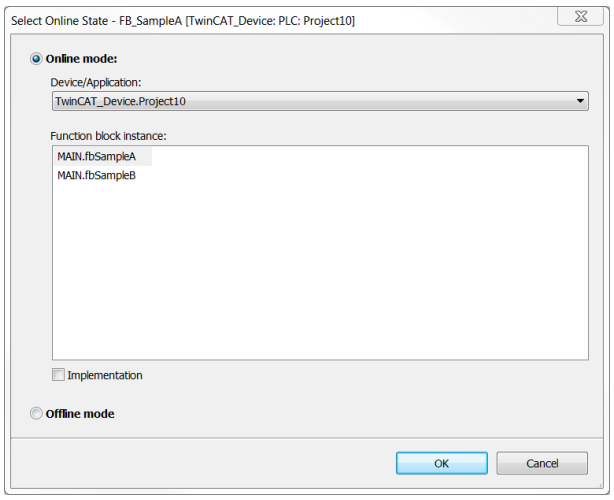
Requirement: An object is selected in the PLC project tree.

In online mode, the dialog **Select Online State** opens, in which you can choose the view in which view the object should be opened. The dialog does not open when the object selection is unambiguous. In this case, the object is opened directly in online mode.

Select Online State dialog

Function: The dialog determines how an object (function block, etc.) that was not yet open in offline mode is to be opened in online mode. You can select whether to open an instance or the basic implementation of the object itself (optionally in online or offline mode).

Requirement: The PLC project contains several instances of the selected object.



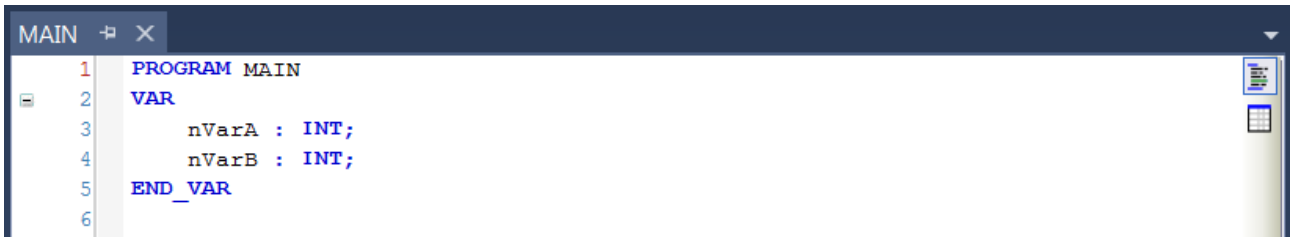
Online Mode	Activate the option to obtain a view in online mode.
Device/Application	Shows the application (project) to which the object is assigned.
Function block instance	If the object is a function block, a list of all instances currently used in the application appears.
Implementation	Select this option to display the basic implementation of the function block, regardless of the selected instance. This option has no function for non-instantiated objects.
Offline mode	Activate the option to obtain a view in offline mode.

5.3.2 Command Textual view

Symbol: 

Function: The command opens the declaration editor in text view.

Call: Button at the right edge of the editor



See also:

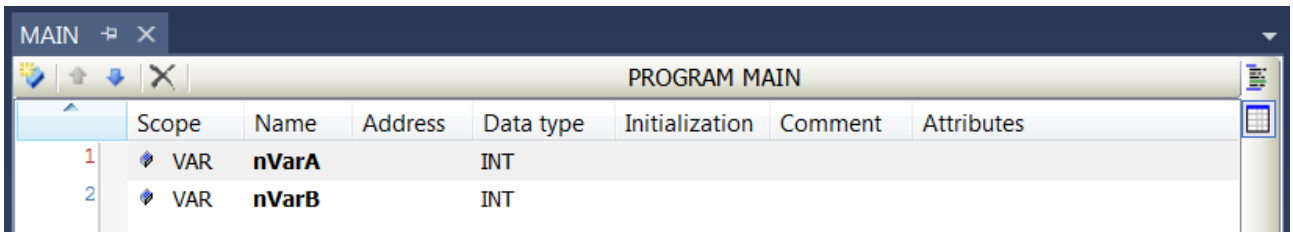
- PLC documentation: Using the Declaration Editor

5.3.3 Command Tabular view

Symbol: 

Function: The command opens the declaration editor in tabular view.

Call: Button at the right edge of the editor



See also:

- PLC documentation: Using the Declaration Editor

5.3.4 Command Full screen

Symbol:

Hotkey: **[Ctrl] + [Shift] + [F12]**

Function: The command switches the TwinCAT display to full-screen mode.

Call: View menu

When you enable the command, the main window of the TwinCAT user interface is displayed in full screen mode. You can return to the preset size by disabling the command again.

5.3.5 Command Toolbars

Function: The command opens a menu for selecting the displayed toolbars.

Call: View menu, Toolbar area context menu

In the menu that opens, select the toolbars you want to show or hide. The command works as an option, i. e. when a toolbar is displayed, it appears in the menu with a check mark in front of it.

See also:

- TC3 User Interface documentation: Customizing Toolbars [▶ 35]

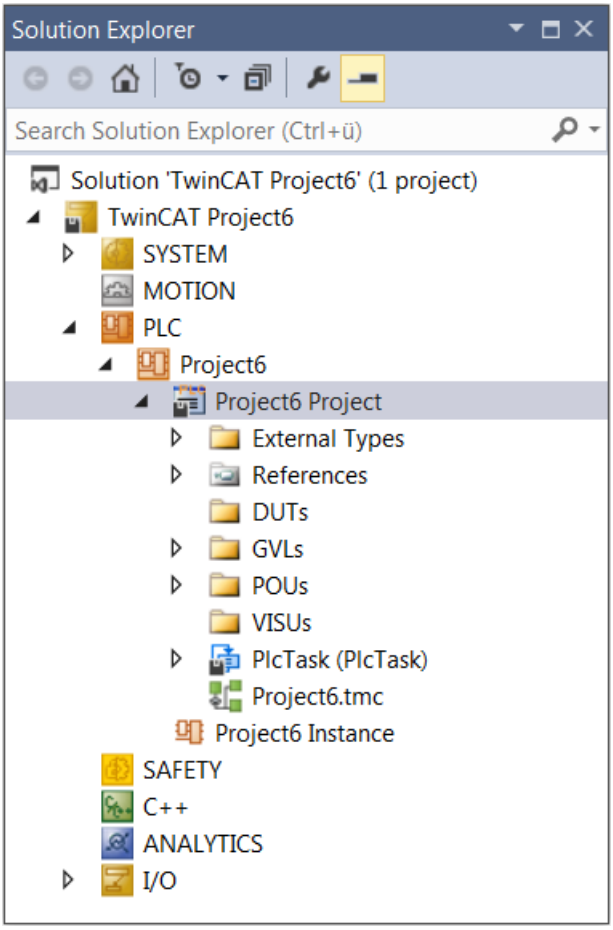
5.3.6 Command Solution Explorer

Symbol:

Function: The command opens the **Solution Explorer** view.

Solution Explorer view

The **Solution Explorer** view shows the TwinCAT 3 project with the corresponding project elements in a structured form. In this view, you can open objects for editing and configuring.



5.3.7 Command Properties Window

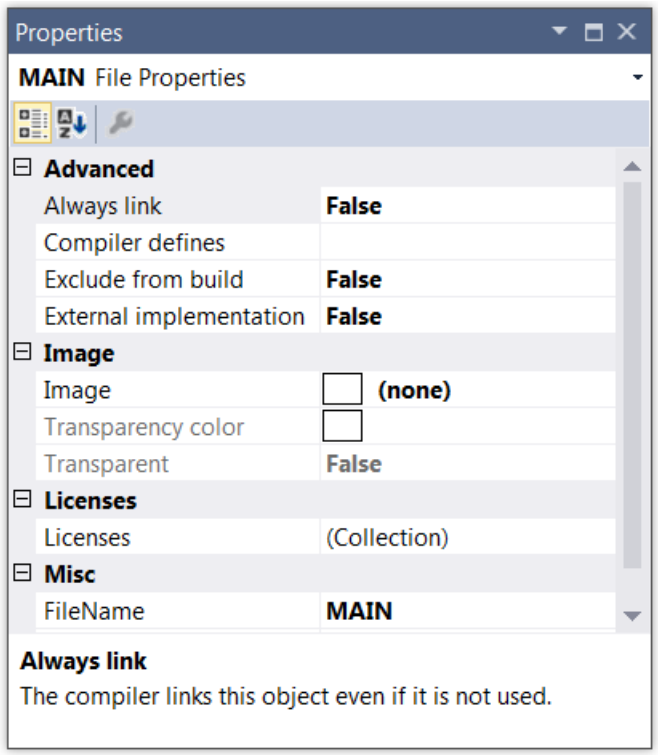
Symbol:

Function: The command opens the **Properties** view.

Call: View menu

Properties view

The **Properties** view shows the properties for the object currently selected in the Solution Explorer. By default, the element properties are sorted by category in a table. Clicking on the plus or minus sign in front of the category to show or hide the associated parameters. A mouse click on the value field of a parameter activates the input mode, in which you can edit the value or property. You can filter or sort the Properties view.



You can also call the Properties window from the context menu of an object in the PLC project tree. A description of the command as well as various object properties can be found in section Command Properties (object) [▶ 90].

5.3.8 Command Toolbox

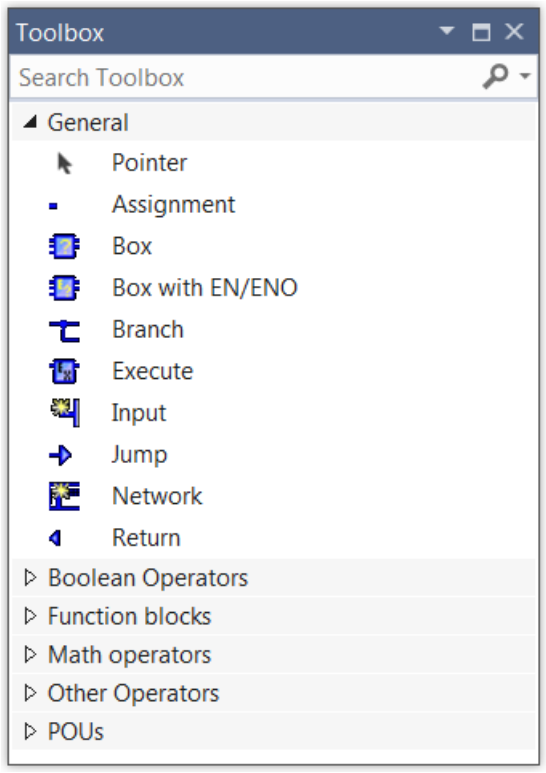
Symbol: 

Function: The command opens the **Toolbox** view.

Call: View menu

Toolbox view

The **Toolbox** view shows the existing tools for the currently active editor. This view is available by default with a graphic editor or a visualization. It contains the graphical programming elements, which you can drag-and-drop into the editor window.



5.3.9 Command Error List

Symbol: 

Function: The command opens the **Error List** view.

Call: View menu

Error List view

The **Error List** view displays errors, warnings and messages related to syntax checking, compile process (compile errors, code size), import processes or Library Manager. The messages are displayed in tabular form.

Filter	Drop-down list for selecting the set of code files to be used: Open Documents Current Project Current Document
Message categories • : Error • : Warning • : Information	Click the message category symbol to show or hide messages. Next to each symbol, TwinCAT displays the number of messages that have occurred.
Clear	Clears the message display
	Clears the filter settings of the error list
Severity level (message category)	Message text with the causing object and the position within the object. Double-click a message entry in the table to go to the source text position.
Code	
Description	
Project	
File	
Line	

Commands in the context menu

Clear	Clears the message display
Show columns	Adds additional columns, which may describe an error in more detail
Copy	Copies the selected error message
Next Error	Selects the next message. The source text position of the next error is displayed.
Previous Error	Selects the previous message. The source text position of the previous error is displayed.

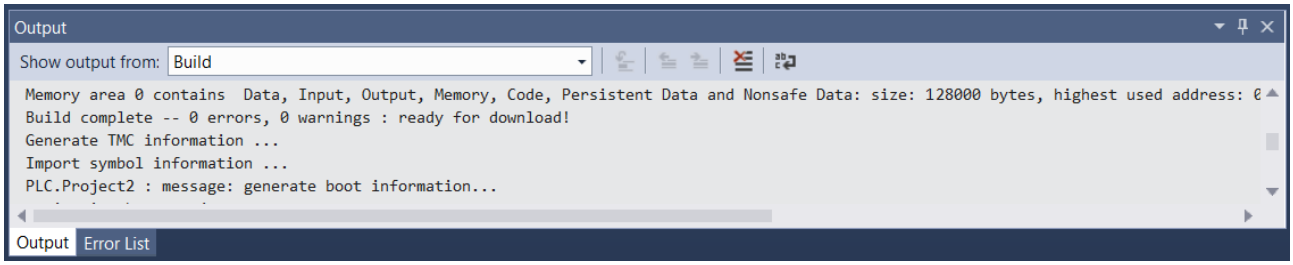
5.3.10 Command Output

Symbol:

Function: The command opens the **Output** view.

Call: View menu

Output view



Message category	The messages are categorized by component or functionality and are available in a selection dialog. You can filter the message display by selecting a category.
 Find the message in the code	The source text position of the message is displayed. Requirement: A message is highlighted.
 Go to the previous message	The previous message is selected.
 Go to the next message	The next message is selected.
 Delete all	Deletes all messages
 Toggle line break	Line break is enabled or disabled

Commands in the context menu

 Copy	Message text is copied
 Delete all	Deletes all messages
 Go To Location	The source text position of the message is displayed. Requirement: A message is highlighted.
 Go to next position	The next message is selected.
 Go to previous position	The previous message is selected.

5.4 Project

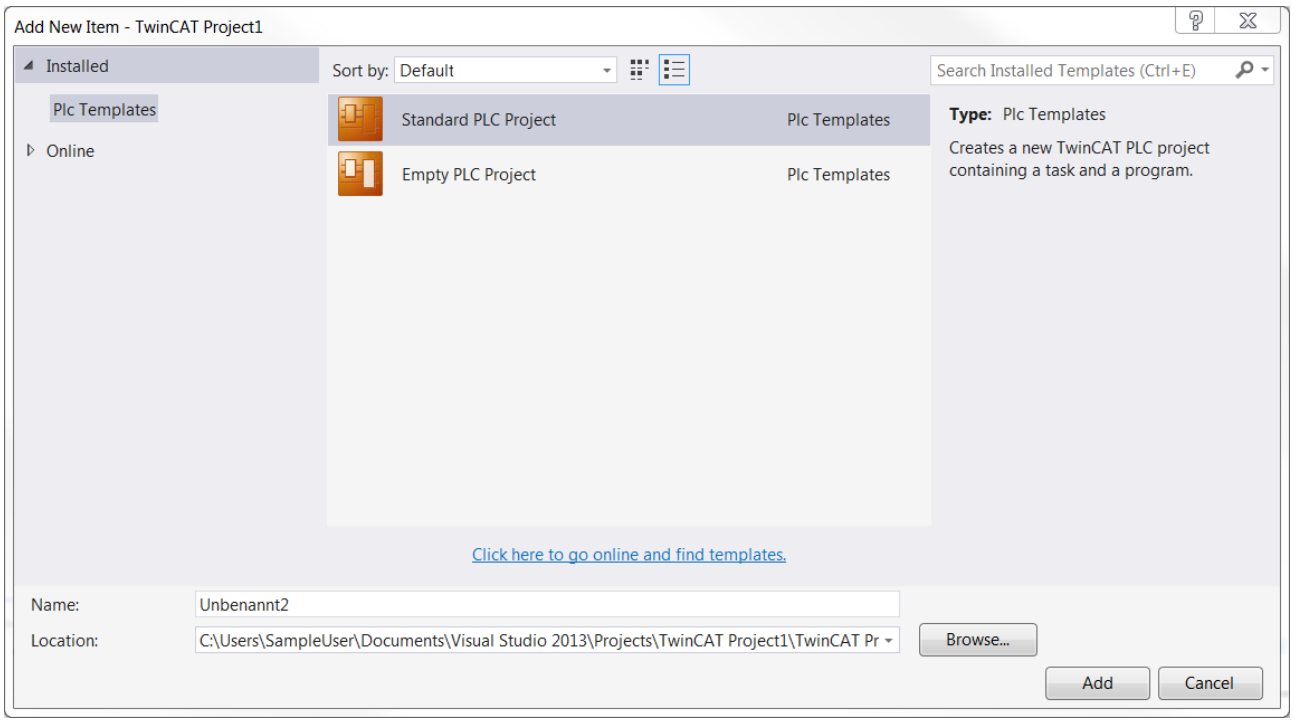
5.4.1 Command Add New Item (project)

Symbol: 

Function: The command opens the **Add New Item** dialog, through which you can create a new PLC project file. (The command is only available if a PLC node is selected.)

Call: Project menu or PLC object context menu in the **Solution Explorer**

Requirement: The PLC node is selected in the TwinCAT project tree.



Plc Templates	Select one of the listed templates. The template determines the basic configuration of a PLC project file. The following templates are available by default: • Standard PLC Project: Creates a new TwinCAT PLC project (*.project). The project is supported by a wizard and contains a Library Manager, a POU "MAIN" program and a referenced task. • Empty PLC Project: Creates an "empty" TwinCAT PLC project for library projects (*.library). The project contains no objects or devices.
Name	Define the name of the new project here. The default name depends on the selected template (usually "Unnamed<n>") and contains a sequential number to ensure that the project name is unique in the file system. You can change the default name according to the file path conventions of the local operating system. A file extension (e.g. .project) can be added. By default, the selected template automatically adds the appropriate extension.
Location	Specify the location for the new project file. The default path depends on the selected template. You can either use the **Browse**... button to open the default browser and specify a path, or you can use the corresponding drop-down list to select a previously entered path.
Add	Clicking **Add** creates a new project based on your settings. If the cursor is placed on an error symbol, a tooltip provides information on how to proceed. If another PLC project is already open, a dialog opens asking you if you want to save and close the project before the new project is opened. The name of the new project is then displayed in the title bar of the TwinCAT XAE frame window. An asterisk ("*") after the name indicates that the project has been modified since it was last saved.

See also:

- PLC documentation: Your first TwinCAT 3 PLC project
- PLC documentation: Creating and configuring a PLC project

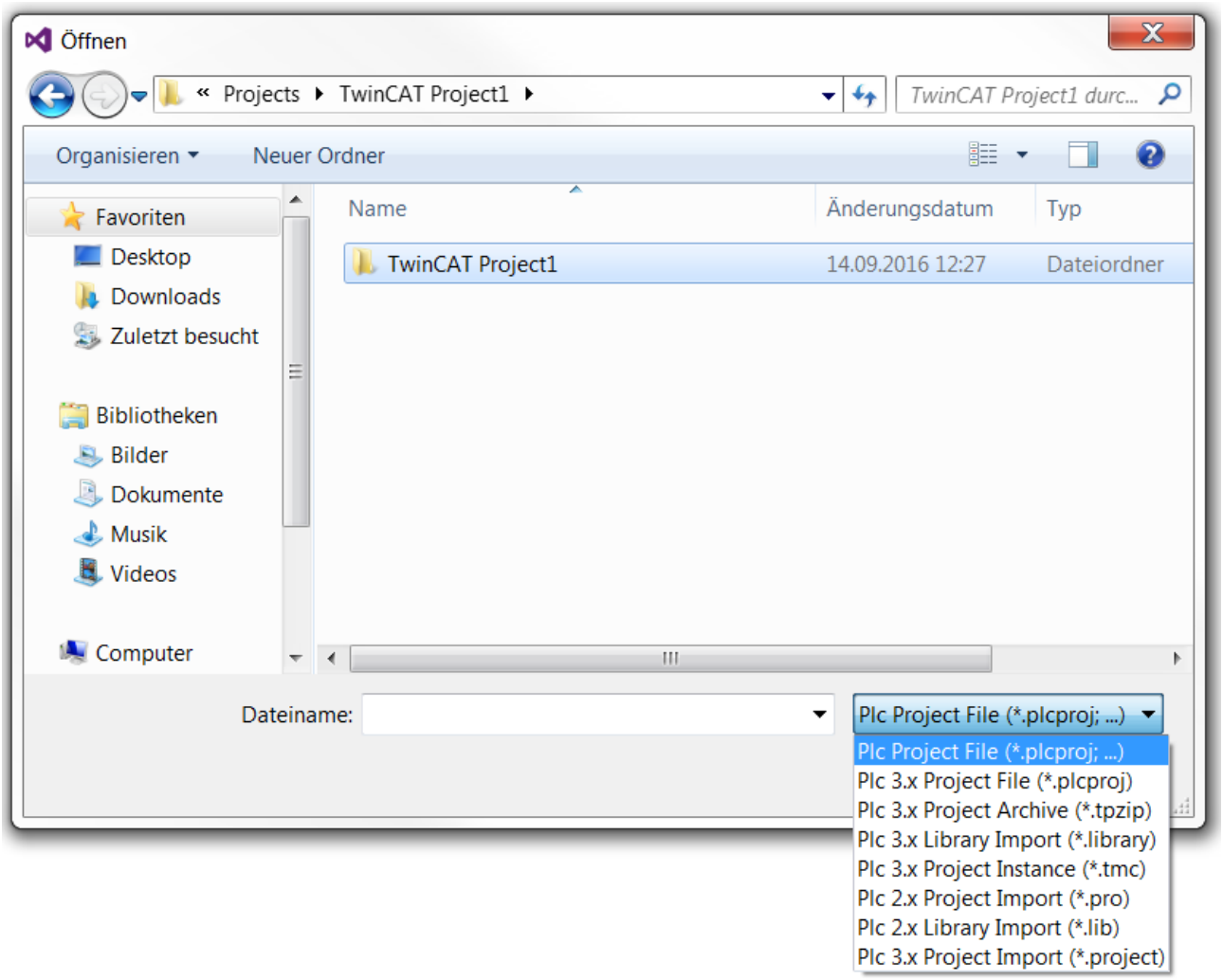
5.4.2 Command Add Existing Item (Project)

Symbol: 

Function: The command opens the standard browser dialog, which can be used to search for a PLC project file and open it in the programming system. If a suitable converter is installed, you can open projects in a different format.

Call: Project menu or PLC object context menu in the **Solution Explorer**

Requirement: The PLC node is selected in the TwinCAT project tree.



File type	By default, you can set the filter to one of the following file types: • PLC 3.x Project file (*.PLCproject): TwinCAT 3 PLC projects with the extension ".PLCproject" • PLC 3.x Project archive (*.tpzip): TwinCAT 3 PLC project archives with the extension ".tpzip" ◦ See also: Command Save <PLC project name> as archive... [▶ 52] • PLC 3.x Library import (*.library): TwinCAT 3 PLC libraries with the extension ".library" • PLC 2.x Project file (*.pro): TwinCAT 2 PLC projects with the extension ".pro" • PLC 2.x Import library (*.lib): TwinCAT 2 PLC libraries with the extension ".lib" • PLC 3.x Project import (*.PLCproject): PLC projects with the extension ".project"
Open	The selected project file is opened or converted and then opened.

***.tpzip PLC project archive**

Content of *.tpzip	The *.tpzip archive folder contains the PLC project to be archived.
Command for creating	A tpzip archive can be created with the following command: Command Save <PLC project name> as archive... [▶ 52]
Note on PLC projects	The files and folders stored in the archive folder for the PLC project depend on the PLC project settings of this PLC project. Settings tab [▶ 112]

Possible scenarios when opening a PLC project

The following scenarios are possible when you open a project:

1. Another project is still open. [▶ 88]
2. The project was saved with an older version of TwinCAT 3. [▶ 88]
3. The project was not saved with TwinCAT 3. [▶ 88]
4. The project was not terminated properly and "Save automatically" was enabled. [▶ 90]
5. The project is read-only. [▶ 90]
6. It is a library that is installed in a library repository and retrieved from it. [▶ 90]

1. Another project is still open.

You are asked if the other project should be saved and closed.

2. The project was saved with an older version of TwinCAT 3.

If the file format differs because the open project was saved with an older version of TwinCAT 3, there are two options:

- If the project cannot be saved in the format of the currently used programming system, you must update it to continue working on the project. The expression that appears at this point: **The changes you made...** refers to internal tasks of various components while the project is loaded.
- If the project can still be saved in the previous format, you can decide whether to update or retain the format. If you decide to retain the format, data loss may occur. If you decide to update the format, the project can no longer be opened with the old version of the programming system.

In addition to the file format, the versions of the explicitly inserted libraries, the visualization profile and the compiler version of the opening project may differ from those installed with the current programming system.

If newer versions are installed on the current programming system, the **Project Environment** dialog opens automatically, where you can update the versions. If no update is made at this point, this can be done later at any time via the **Options > Project Environment** dialog.

Note the compiler version

If a project is opened that was created with an older version of the programming system and for which the latest compiler version is set in the project settings, while the project environment setting for the compiler version is set to **Do not update** in the new programming system, the compiler version that was used last in the old project continues to be used (i.e. not the "Current" version in the new environment).

3. The project was not saved with TwinCAT 3.

Case 1)

If you set the file filter when selecting the project to be opened and an appropriate converter is available, the converter is used automatically and the project is brought into the current format. The conversion is converter-specific. Usually, you are prompted to define the handling of referenced libraries or device references.

TwinCAT 3 converter

Adaptation of a TwinCAT PLC control project to the TwinCAT 3 syntax can only be successful during import if the converter is able to compile the project without errors.

If you have set the **All Files** option when selecting the project to be opened, no converter is enabled and the **Convert Project** dialog opens. In the dialog, you need to explicitly trigger the conversion of the project by selecting one of the options.

- **Convert to the current format:** Select the converter you want to use from the selection list (application for conversion). After the conversion, the project can no longer be opened in the old version.
- **Create a new project and add a specific device:** (Not yet implemented)

TwinCAT 2.x PLC Control project options

The project directory path set in the TwinCAT 2.x PLC control project options and the project information are adopted in the **Project Information** dialog.

Case 2)

If libraries are integrated in the project for which "conversion mapping" has not yet been stored in the library options, the **Converting a library reference** dialog appears, in which you can define how this reference should be converted:

- **Convert and install the library:** If you select this option, the referenced library is converted to the new format and remains referenced in the project. It is automatically installed in the library repository under the **Other** category and continues to be used. If the library does not have the project information (title, version) required for an installation, you will be prompted to enter it in the **Enter Project Information** dialog.
- **Use the following library, which is already installed:** If you select the options, the referenced library is replaced by another one that is already installed on the local system. Use the **Select** button to open the **Select...** dialog. Here you can select the desired version of one of the installed libraries. This corresponds to the configuration of the version handling in the **Library Properties** dialog. An asterisk ("*") means that, as a rule, the latest version of the library available on the system is used in the project. The list of available libraries is structured in the same way as in the **Library Repository** dialog. You can sort the list by company and category.
- **Ignore the library. The reference will not appear in the converted project:** If you enable this option, the library reference is removed. The library is then no longer included in the converted project.
- **Use this mapping in future if this library is present:** If you enable this option, the settings made here in the dialog will also be applied to future project conversions, as soon as the respective library is referenced.

In the converted project, the library references are defined in the Global Library Manager in the Solution Explorer. After the conversion of the library references, the project conversion continues with the **Open Project** dialog, as described above.

For general information on library management, see section “Using libraries” in the PLC documentation.

Case 3)

When you open a TwinCAT 2.x PLC Control project that references a device (target system) for which no "conversion mapping" has yet been defined in the TwinCAT 2.x PLC Control converter options, the **Device Conversion** dialog opens, in which you can specify whether and how the old device references are to be replaced by more recent ones. The device originally used is displayed. Choose one of the following options:

- **Use the following already installed device:** Click the **Select** button to open the **Select target system** dialog, in which you can select one of the devices currently installed on the system. This device is then inserted in the **Solution Explorer** of the converted project, instead of the old one. Select the option **Select a target system...** to select one of the devices listed. The list of available devices is structured in the same way as in the **Device Repository** dialog. You can sort the list by manufacturer or by category.
- **Ignore the device. No application-specific objects will be available:** If you enable this option, no entry for the device is created in the **Solution Explorer** of the new project, i.e. the device is ignored during the conversion, and no application-specific objects, such as the task configuration, are applied.

- **Save this assignment for future reference:** If you select this option, all settings of the dialog, i.e. the displayed "conversion mapping" for the device, are saved in the TwinCAT 2.x PLC Control Converter options and applied to future conversions.

4. The project was not terminated properly and "Save automatically" was enabled.

If the **Auto Save** function was enabled in the **Load and Save** options and TwinCAT 3 PLC was not terminated regularly after the last modification of the project without saving, the **Auto Save Backup** dialog opens for handling the backup copy.

5. The project is read-only.

If the project to be opened is read-only, you are asked whether you want to open the project in write-protected mode or whether you want to unlock it.

6. It is a library that is installed in a library repository and retrieved from it.

An error message is displayed if you try and open a library project that is installed in a library repository. You cannot open a library project using this path. After closing the dialog with **OK**, the project name appears in the title bar of the user interface. An asterisk ("*") after the name indicates that the project has been modified since it was last saved.

See also:

- PLC documentation: Open a TwinCAT 3 PLC project
- PLC documentation: Open a TwinCAT 2 PLC project

5.4.3 Command Properties (object)

Function: This command enables the **Properties** view, which displays general information about the currently selected object.

Call: PLC object context menu

Requirement: An object is selected in the PLC project tree.

Depending on the object currently selected, the following property areas are displayed:

- Advanced [▶ 90] (compile settings)
- Image [▶ 91]
- Licenses [▶ 91]
- General [▶ 91] (object name, object path)
- SFC Settings [▶ 94] (flags for Sequential Function Chart)

The special visualization properties are documented under Visualization object, and the special library and placeholder properties are documented under Command Properties.

Advanced

This area shows the settings for compiling the object.

Advanced	
Always link	**False**
Compiler defines	
Exclude from build	**False**
External implementation	**False**

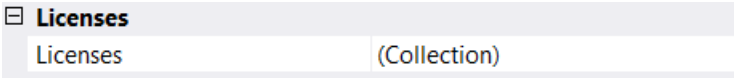
Always bind	True: The object is selected in the compiler and therefore always included in the compile information. It is therefore always compiled and loaded onto the PLC. This option becomes relevant if the object is located below an application or referenced via libraries that are also located below an application. The compile information is also used as a basis for the selectable variables of the symbol configuration.
Compiler definitions	The compiler definitions entered here are currently not evaluated. If you want to use compiler definitions, enter them in the PLC project properties. See: • Command properties (PLC Project) > Category Compile [▶ 97] • PLC documentation: Reference Programming > Pragmas > Conditional pragmas
Exclude from compilation	True: The object is not included in the next compilation run.
External implementation	(late binding in the runtime system) True: No code is generated for this object when the project is compiled. The object is not linked until the project is loaded to the target system, provided it exists there (in the PLC runtime system or in another real-time module).

Image

In this area you can assign an image to the object, which is displayed in the graphical view of the library manager and in the toolbox of the FBD/LD/IL editor. Transparency of the image can be achieved by selecting a color, which is then displayed transparently. If you select the option **Transparency Color**, you can use the rectangular button to the right of it to open the standard dialog for selecting a color.

Licenses

This section contains a list of licenses for the object.



General

In this section you will find general information about the selected object.

FileName	File/object name
FullPath	Storage path/location of the object (not editable at this point)
Version	File version, values: • 1.1.0.1: This file version is used when the object is saved in XML format. • 1.2.0.0: This file version is used when the object is saved in Base64 format. Please note that objects with file version 1.2.0.0 (or higher) cannot be loaded with Engineering versions lower than TC3.1.4024! (not editable at this point, indirectly configurable via the storage format, see **Format** property)

Engineering incompatibility of file version 1.2.0.0 (or higher) with TC3.1 < build 4024

Please note that objects saved with file version 1.2.0.0 (or higher) cannot be loaded with Engineering versions lower than TC3.1.4024!

Since an object is automatically saved with file version 1.2.0.0 when using the optional Base64 format, objects with Base64 format cannot be loaded with Engineering versions lower than TC3.1.4024.

If a PLC project contains objects with the file version 1.1.0.1 and objects with the file version 1.2.0.0, the 1.1.0.1 objects are loaded with an Engineering version lower than TC3.1.4024. Objects with file version 1.2.0.0 are not loaded.

The file version of a file saved with file version 1.2.0.0 can be reset to 1.1.0.1 using XAE version TC3.1.4024 or higher.

Options

In this section you will find some options that can be configured for PLC objects.

Format	**Individual setting option of the storage format:** The storage format of an object can be individually configured at this point for the object types listed below. Storage format, values: • XML: The object is saved in XML format. ◦ Objects with this storage format are stored in file version 1.1.0.1, see **Version** property. • Base64: The object is saved in Base64 format. ◦ Objects with this storage format are stored in file version 1.2.0.0, see **Version** property. Please note that objects with file version 1.2.0.0 (or higher) cannot be loaded with Engineering versions lower than TC3.1.4024! **Advantages of Base64 over XML:** Base64 results in compressed storage, compared to XML. As a result, improved performance can be achieved with file access to these objects, which can be used, for example, when loading, moving or copying objects. **Availability of Base64:** The Base64 storage format is optionally available from build 4024 for the following PLC objects: • POUs where the POU body is programmed in a graphical implementation language ◦ Sequential Function Chart (SFC) ◦ FBD/LD/IL (Function Block Diagram/Ladder Diagram/Instruction List) ◦ CFC (Continuous Function Chart and page-oriented CFC) ◦ UML class diagram and Statechart • POUs with a subelement (e.g. action, method) that is programmed in a graphical implementation language (for graphical languages see first key point) • Visualizations • Visualization Manager • Text lists • Recipe manager • Image pool **Setting option for the standard storage format:** For a PLC project, the setting "Write object content as" in the PLC project properties (Category Advanced [▶ 107]) can be used to define the standard storage format for the object types mentioned above.
Separate LineIds	Value: True or False • True: The line IDs of this POU are stored in a separate file (LineIDs.dbg). • False: The line IDs of this POU are stored in the POU itself. (not editable at this point, configurable in the Write options [▶ 171])
Sort	Value: Name or GUID Specifies the sequence in which the child objects (such as methods) are stored in the parent object: either sorted by name or by GUID. (not editable at this point, configurable in the Write options [▶ 171])
Write ProductVersion	Value: True or False (not editable at this point, configurable via the setting "Write product version in files" in the Category Advanced [▶ 107] of the PLC project properties)

SFC settings

This area displays the current settings for compiling and handling implicit variables for the SFC object currently selected.

SFC	
Use default SFC settings	True
SFC Build	
CalculateActiveTransitionOnly	False
SFC Flags	
SFCCurrentStep	Declare
SFCEnableLimit	Declare
SFCError	Declare
SFCErrorAnalyzation	Declare
SFCErrorAnalyzationTable	Declare
SFCErrorPOU	Declare
SFCErrorStep	Declare
SFCInit	Use
SFCPause	Declare
SFCQuitError	Declare
SFCReset	UseDeclare
SFCTip	Declare
SFCTipMode	Declare
SFCTrans	Declare

Use default SFC settings	TRUE (default): With this option, the default values defined in the PLC project properties can be applied to the currently selected object and displayed in the **Properties** view of the object. FALSE: This option allows you to configure SFC settings that are valid specifically for this SFC object.
Compile ("Build") and variables ("Flags")	The meaning of these items corresponds to the standard settings for SFC objects, which are configured in the PLC project properties.

5.4.4 Command Properties (PLC project)

Symbol: 

Function: This command opens an editor window in which the properties of the project and additional project-related information can be displayed and defined.

Call: Context menu of the PLC project object (<PLC project name> Project) or **Project** menu

Requirement: A project is open.

TwinCAT stores the PLC project properties directly in the PLC project.

Scope of PLC project properties

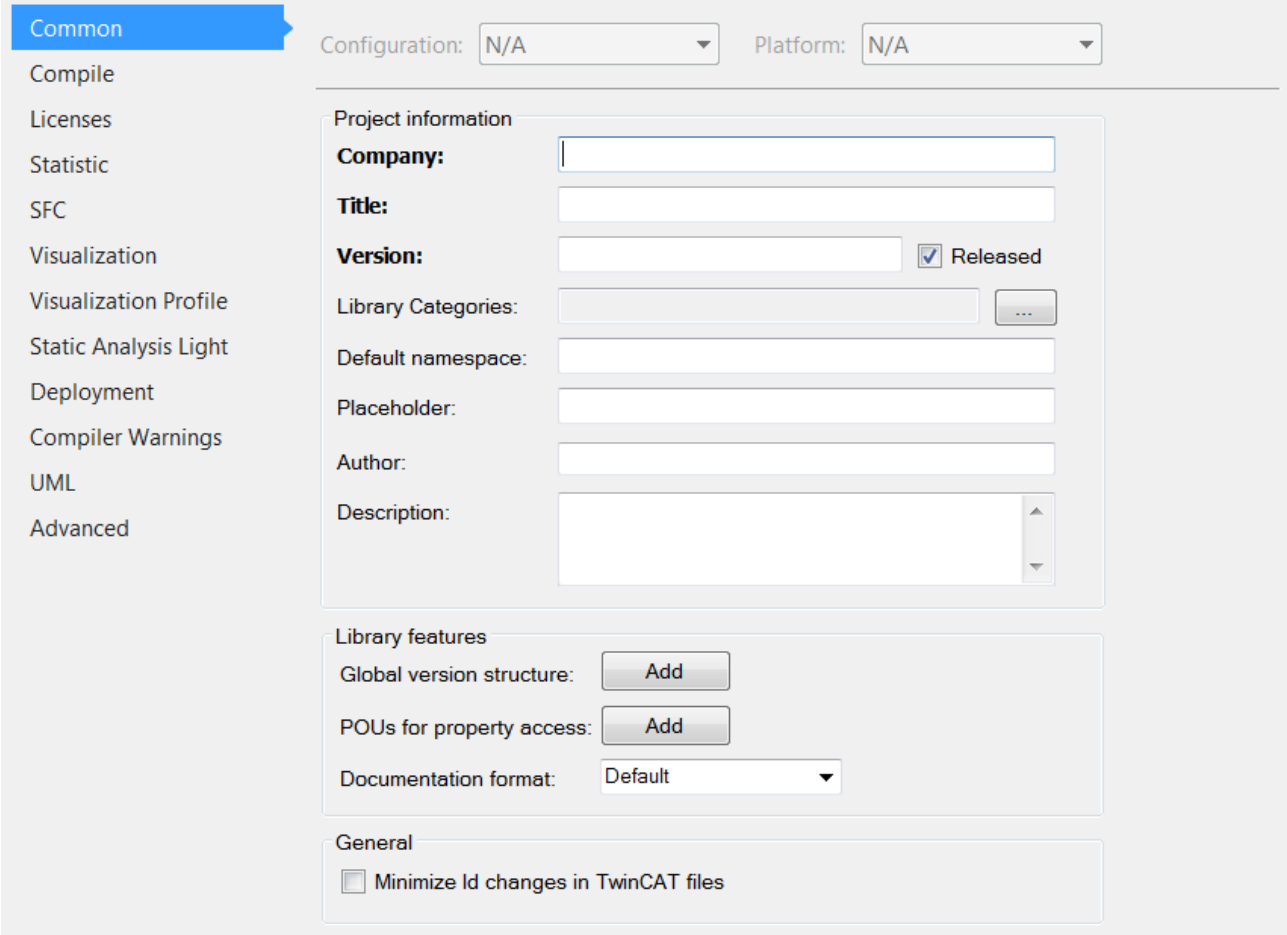
Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

See also:

- PLC documentation: Configuring a project

5.4.4.1 Category Common

The category **Common** contains general information and meta-information of the project file. TwinCAT uses this information to create keys in the **Properties** tab. For example, if the **Company** text field contains the name "Company_A", the **Properties** tab contains the **Company** key with the value "Company_A".



Project information

For a library project, a company, title and version must be entered in order to be able to install the library.	
Company	Name of the company, which created this project (application or library). In addition to the library category, it is used for sorting in the library repository
Title	Project title
Version	Project version, e.g. "0.0.0.1"
Released	 : Protection against modification enabled. Result: When you now edit the project, a prompt will appear asking you if you really want to change the project. If you answer this query once with Yes, the prompt will no longer appear when the project is edited again.
Library Categories:	Categories of the library project by which you can sort in the Library Repository dialog. If no category is specified, the library is assigned the category "Other". To assign it to another category, the category must be defined. Library categories are defined in one or several external description files in XML format. To assign the library, you can either call such a file or another library file that has already picked up information about the categories from a description file. Requirement: The project is a library project.
	The **Library Categories** dialog opens, in which you can add library categories.
Default namespace:	The default setting for the namespace of a library is the library title. Alternatively, a different namespace can be defined explicitly, either generally for the library at this point in the project information during library generation, or in the **Properties** dialog of the library reference for local use of the library in a project. The namespace of the library must be used as prefix of the identifier, in order to enable unambiguous access to a module that exists more than once in the project, or if the use of this prefix is enforced by the library property LanguageModelAttribute "qualified-access-only" ("Unambiguous access to library modules or variables"). If you do not define a standard namespace here, the name of the library file is automatically used as the namespace.
Placeholder	At this point, a default name for the placeholder can be specified, which represents or references this library. If a placeholder is not specified explicitly at this point, the default setting for the placeholder name of a library corresponds to the library title.
Author	Project author
Description	Brief description of the project (e.g. content, functionalities, general information such as "for internal use only", etc.)

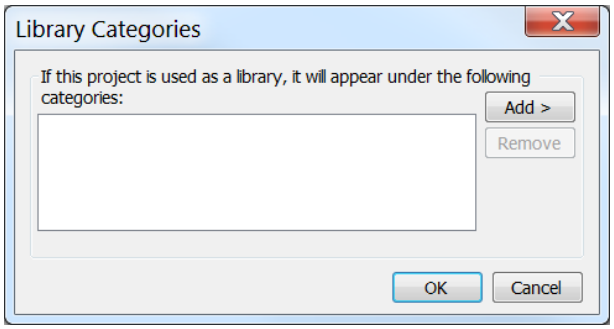
Library features

Creating a global version structure	Creates a global variable list in the PLC project, which contains the version information.
Automatically generate library information POU	**Add** button: POU objects of type "Function" are automatically created in the project tree, which can be used to access project properties in the application program. In this case, special functions are generated for the properties **Company**, **Title** and **Version** (F_GetCompany, F_GetTitle, F_GetVersion). If these functions have been added to the project by clicking **Add**, they can be removed from the project by clicking **Remove**.
Documentation format	Option **reStructuredText**: During library creation, comments that correspond to a specific format are restructured and displayed in the **Documentation** tab of the Library Manager in this customized view. This opens up additional options for library documentation.

General

Minimize ID changes in TwinCAT files	The GUIDs of the PLC objects (e.g. POUs) are linked to those of the PLC project (using XOR). This avoids changes to the GUIDs of the PLC objects when they are used several times in different projects.

Library Categories dialog



List of categories	List of categories assigned to the library project. They can come from several sources. When you have entered all the desired categories, confirm the dialog with **OK**.
Add	The commands **From Description File...** and **From Other Library...** appear.
Remove	TwinCAT removes the selected category.
From description file...	The **Select Description File** dialog appears, in which you can select a description file with the extension *.libcat.xml. The file contains command categories. When you exit the dialog with **Open**, TwinCAT applies these categories.
From another library....	The **Select Library** dialog appears, in which you can select a library (*.library) whose command categories are to be adopted. When you exit the dialog with **Open**, TwinCAT applies these categories.
OK	TwinCAT provides the categories as project information and displays them in the **Library Categories** field.
Cancel	Closes the dialog. The process is aborted.

See also:

- PLC documentation: Configuring a project
- PLC documentation: Using libraries

5.4.4.2 Category Compile

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring. Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **Compile** is used to configure the compiler options.

Settings

Compiler definitions	Here you can enter compiler definitions/"defines" (see {define} statements) and conditions for compiling the application (conditional compilation). A description of the available conditional pragmas can be found in section Conditional pragmas. The expression expr used in such pragmas can also be entered here. Several entries are possible in the form of a comma-separated list.
System compiler definitions	Available from TC3.1 Build 4024 The compiler definitions that have been set at System Manager level in the PLC project settings under Compiler definitions [▶ 111] are automatically adopted here.
Download application info	Available from TC3.1 Build 4024 Situation: You are loading a PLC project onto the controller that differs from the project already located there. In this case, a message window appears containing a **Details** button. Use this button to open the **Application information** window, which allows you to check the differences between the current PLC project and the PLC project on the controller. This involves comparing the number of function blocks, the data and the storage locations. The **Application information** window contains a brief description of the differences, for example: • Declaration of MAIN changed • Variable fbMyNewInstance inserted in MAIN • Number of methods/actions of FB_Sample changed  (Default): If this setting is enabled, the information on the contents of the PLC project is loaded onto the PLC. This enables an extended check of the differences between the current PLC project and the PLC project on the controller. The difference in the extended check option is that the **Application information** window contains an additional **Online comparison** tab, which shows a tree comparison view. This will tell you which POUs have been changed, deleted or added. The additional tab appears when you execute the blue underlined command in the lower area of the **Application information** window ("Application not current. Generate code now to display the online comparison?“).

Generate tpy file	Available from TC3.1 Build 4024 The tpy file contains, among other things, project, routing, compiler and target system information. It is the format used for describing a TwinCAT 2 PLC project. To ensure compatibility with existing applications, this file can be created for a TwinCAT 3 PLC project, if required. ☐ (Default): When the PLC project is created, no tpy file associated with the project is created. ☑: When the PLC project is created, a tpy file associated with the project is created and stored in the project folder. Please note that the value and configuration availability of this option depends on whether or not the TPY file is configured as a target file (see Settings tab [▶ 112]). • If the TPY file is enabled as a target file, the following happens: ◦ TwinCAT remembers the current status of the "Generate tpy file" option (= "original value", see below.). ◦ If this is not already the case, the option "Generate tpy file" is automatically activated next time the project is created. ◦ In addition, the "Generate tpy file" option is grayed out so that it cannot be disabled by the user as long as the TPY file is configured as a target file. • If the TPY file is subsequently disabled as a target file, the following happens: ◦ Next time the project is created, the "Generate tpy file" option is assigned its "original value" (see above.). ◦ In addition, the option is no longer grayed out, making it available again for configuration by the user.

Solution options

Compiler Version	Defines the compiler version that TwinCAT uses during compilation and during loading for compilation. Please note that this setting is not a substitute for the Remote Manager. If the PLC project is an application project, the Remote Manager should always be used for handling different Engineering versions. In this case, the compiler version should always be set to "latest". The compiler version setting is only relevant if the PLC project to be version-managed is a library project. It is advisable to save the oldest version of the library that will ultimately be used in practice. To this end, the compiler version must be set to the corresponding fixed version (e.g. "3.1.4018.0").
Maximum number of warnings	Refers to the maximum number of warnings that TwinCAT issues in the **Error List** view. The selection of displayed compiler warnings is defined in the category **Compiler warnings** in the **Project Settings** dialog.
Replace constants	☑: TwinCAT loads the value directly for each constant of scalar type, i.e. not for STRING, ARRAY or structures. In online mode, TwinCAT identifies the constants in the declaration editor or monitoring window with a symbol preceding the value. In this case, access via an ADR operator, forcing or writing, for example, is not possible. ☐ (Default): Access to constants is possible. The computing time increases slightly.

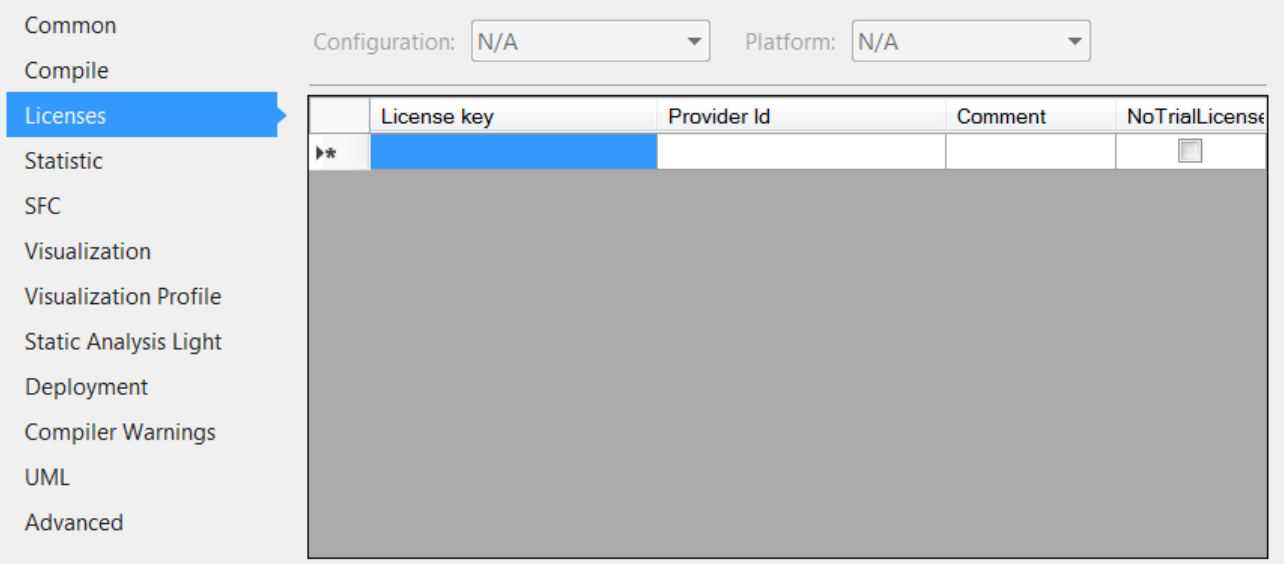
See also:

- Category Compiler Warnings [▶ 106]

5.4.4.3 Licenses category

In future, the **Licenses** category is intended to facilitate allocation of a TwinCAT 3 OEM license to a custom or proprietary library. This feature is not yet implemented.

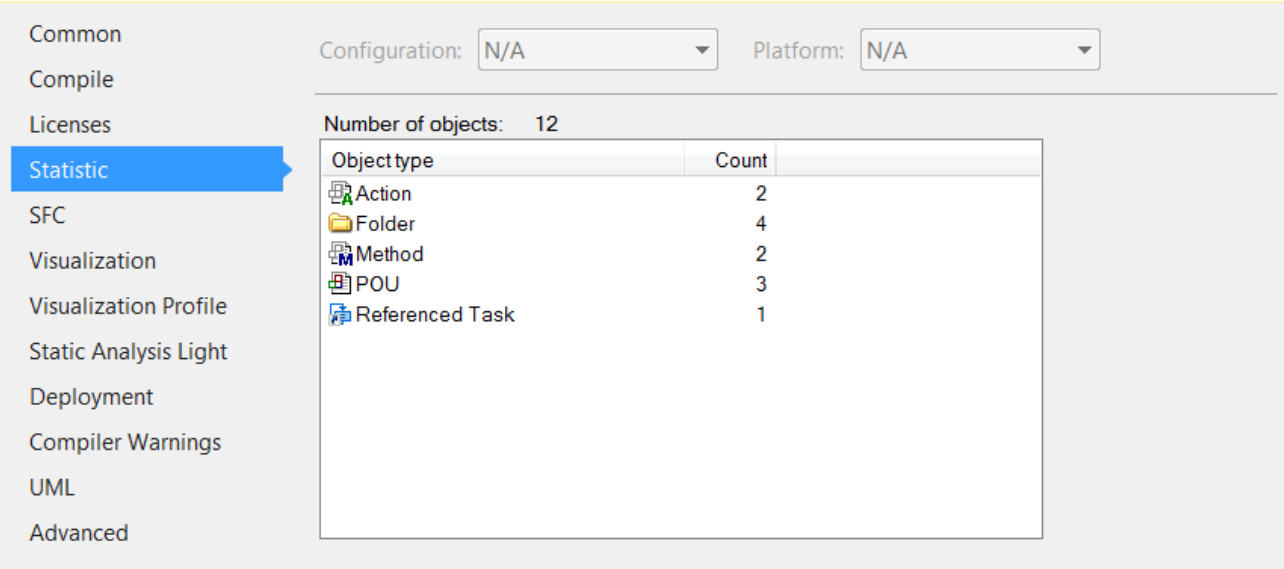
This category is therefore not yet supported in the current TwinCAT version (reserved for future use).



The user must currently query an OEM license for his own library in the code of the library. See Query of an OEM license in the PLC application.

5.4.4.4 Category Statistic

The category **Statistic** provides statistical information on how many objects of the different types are present in the project.



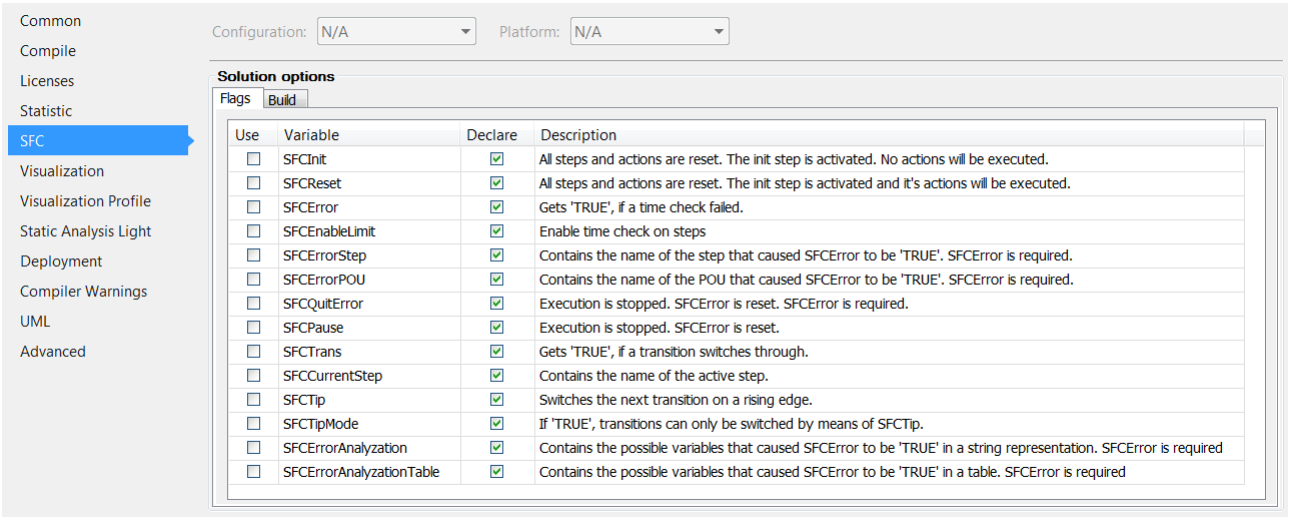
5.4.4.5 Category SFC

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **SFC** is used to configure the settings for SFC objects. Each new SFC object automatically has the configured settings in its properties.

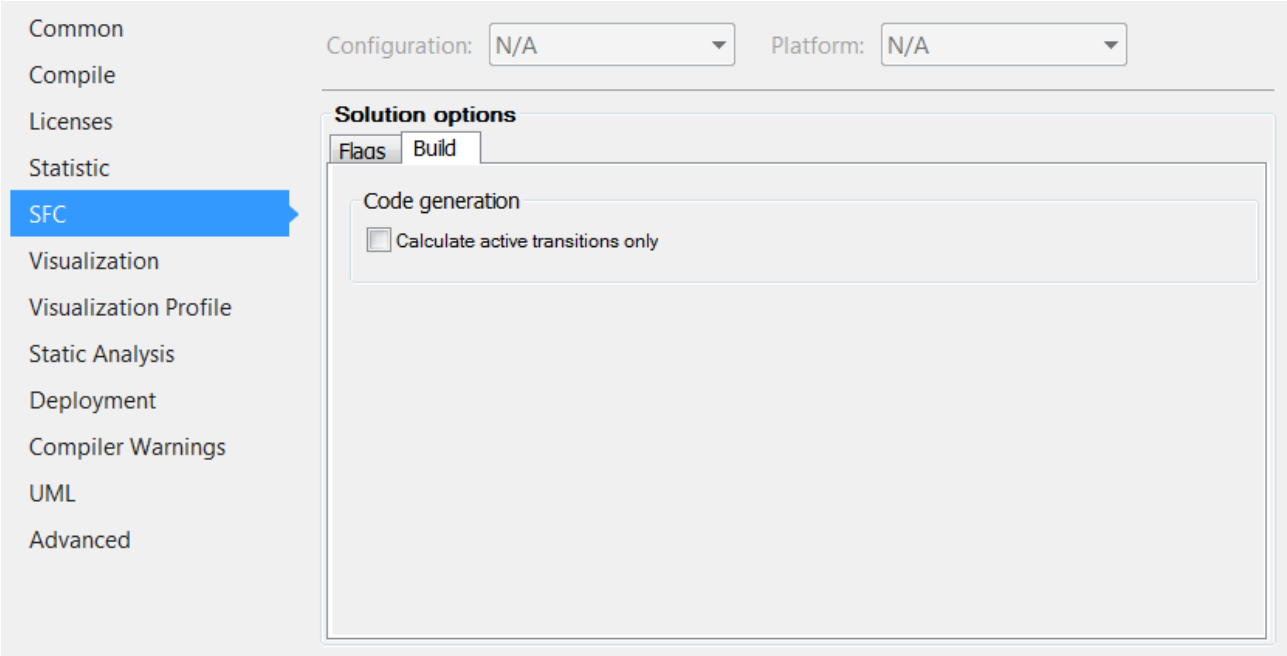
Flags tab



Implicitly generated variables (flags) for controlling and monitoring the processing in an SFC diagram.	
Use	: The corresponding variable is used.
Declare	: The corresponding variable is created automatically. Otherwise, if the usage is intended (Use is set), the user has to declare the variable.

An automatically declared flag variable appears in the declaration part of the SFC editor, but only in online mode.

Build tab



Code generation

Calculate active transitions only	: TwinCAT generates code only for transitions that are currently active.

See also:

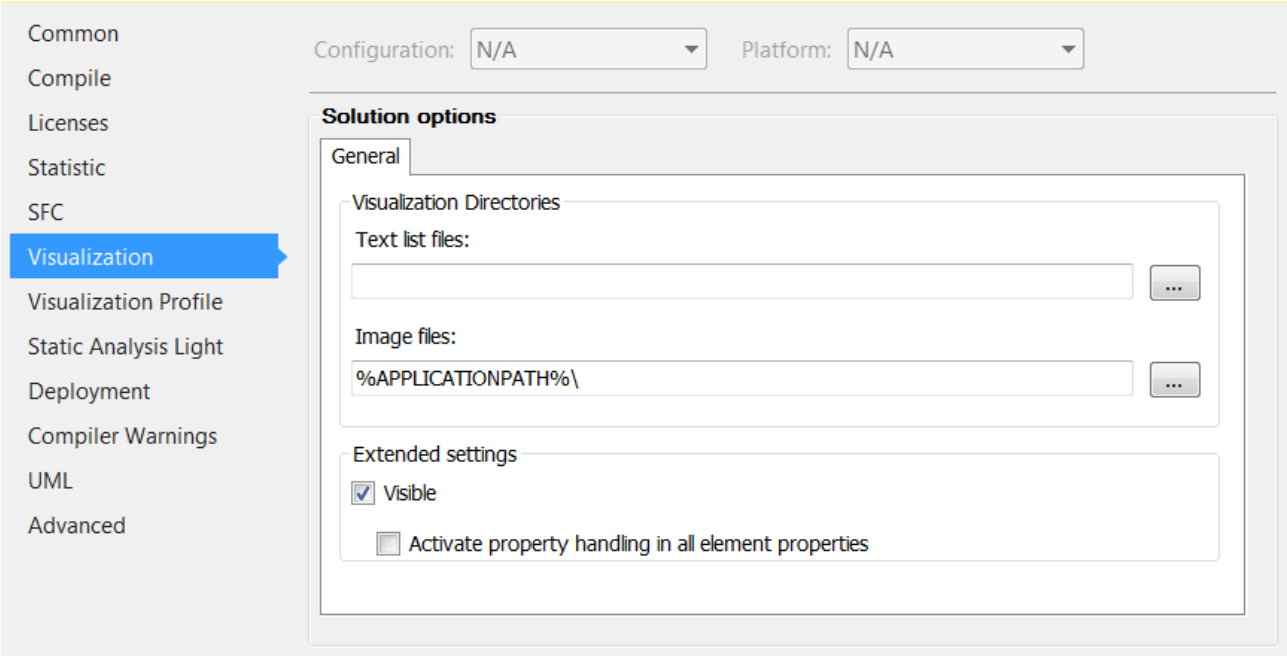
- SFC Flags

5.4.4.6 Category Visualization

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring. Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **Visualization** is used to configure the project-wide settings for objects of type Visualization.



General tab

Visualization Directories

Text list files	Directory containing text lists available in the project for configuring texts for different languages. TwinCAT uses this directory when exporting or importing text lists, for example. Click ... to bring up the **Find Folder** dialog, which allows you to select a directory in the file system.
Image files	Directory containing image files available in the project. Multiple folders are separated by semicolons. TwinCAT uses this directory when exporting or importing image files, for example. Click ... to bring up the **Find Folder** dialog, which allows you to select a directory in the file system.

Extended settings

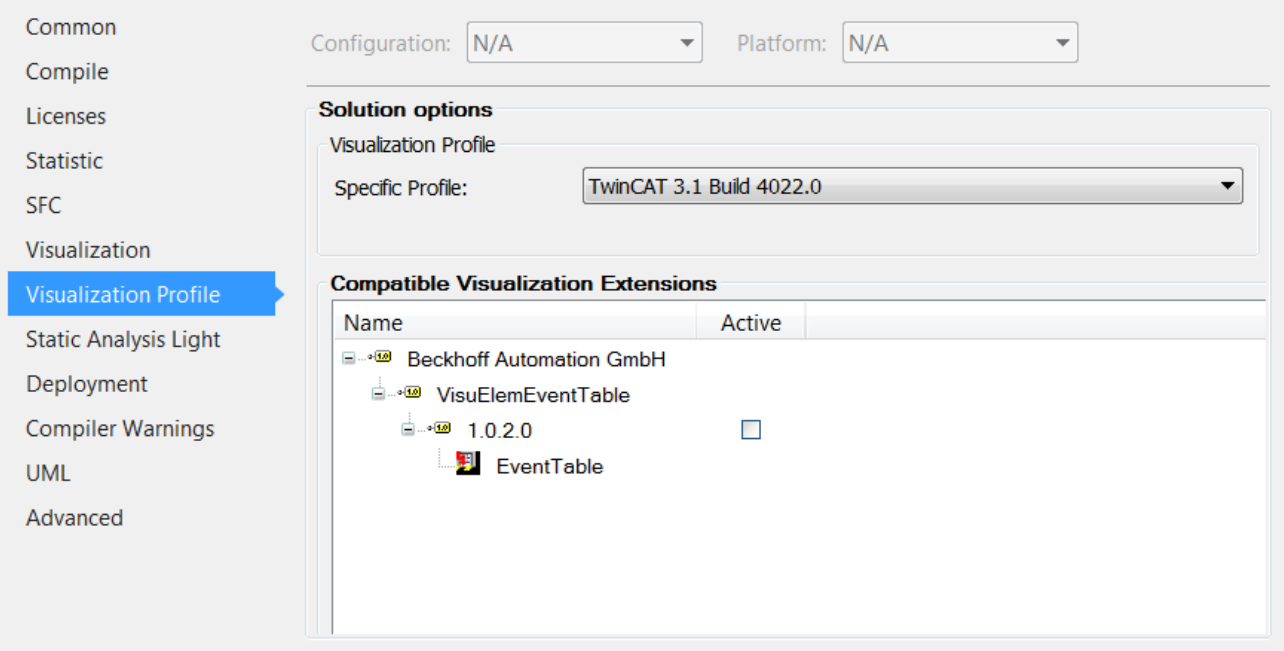
Enables property handling in all element properties	: You can also configure a visualization element in those of its properties, in which you select an IEC variable, with a property. TwinCAT then generates additional code for properties handling when compiling a visualization. Requirement: Your IEC code contains at least one object of type Interface property, i.e. a property . MAIN (PRG) Property_A Get Set Requirement: **Visible** is enabled.

5.4.4.7 Category Visualization Profile

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The **Visualization Profile** category allows you to set the visualization profile.



Visualization profile

Specific Profile	Profile, which TwinCAT uses in the project and which determines the visualization elements that are available in the project. The selection list contains all previously installed profiles.

5.4.4.8 Category Static analysis

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **Static analysis** defines the checks that are taken into account in the static code analysis.

Static Analysis Light:

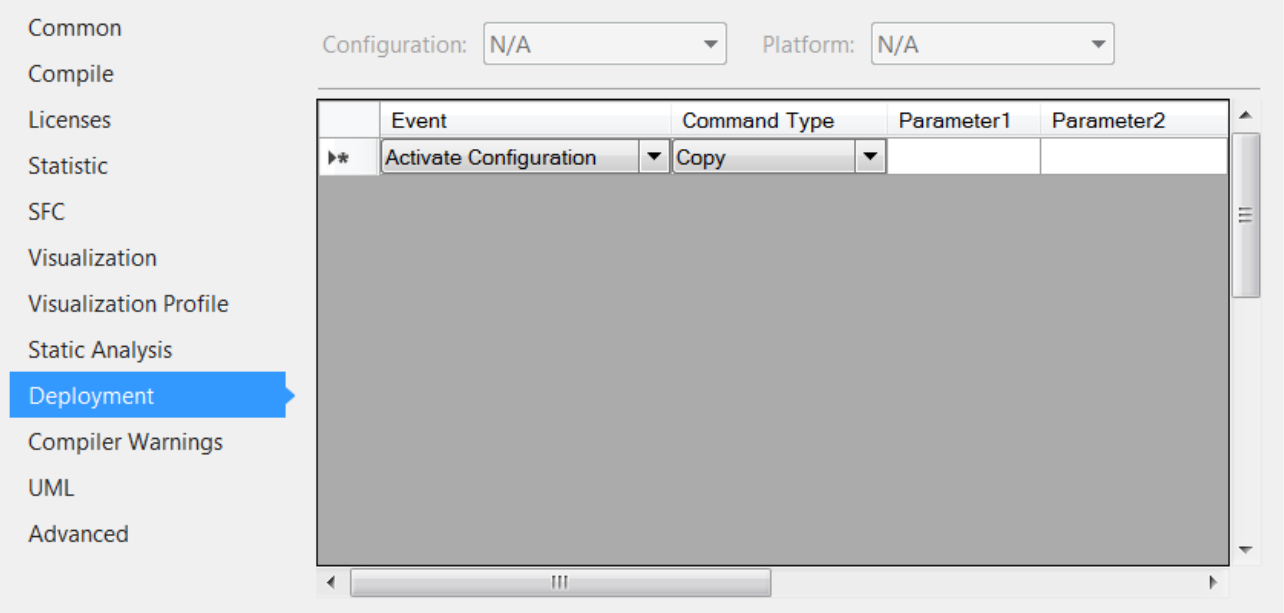
- If you have not activated the additional TE1200 Engineering license, you can use the license-free version of Static Analysis (Static Analysis Light), which contains some coding rules. The free Light version enables you to familiarize yourself with the basic handling of the product, for example, based on a reduced set of functions.
- For more information about Static Analysis Light see:
  PLC documentation: Programming a PLC Project > Checking syntax and analyze code > Code analysis (Static Analysis)

Static Analysis Full:

- If you have enabled the additional TE1200 Engineering license, the full range of Static Analysis functions is available (saving and loading settings, more than 100 coding rules, naming conventions, metrics, forbidden symbols).
- For more information about Static Analysis Full see: TE1200 Static Analysis.

5.4.4.9 Category Deployment

The category **Deployment** is used to set up commands that are to be executed during the installation and startup of an application.



The following events are available, after which the commands listed in the list can be called:

Activate Configuration	The required command is called up after the configuration has been enabled.
Plc Download	The required command is called up after the PLC application has been downloaded to the target system.
Plc Online Change	The required command is called after a successful online change.
Plc After Compile	The required command is called after a compilation of the PLC application.

The following commands can be executed:

Copy	Copies files from parameter 1 (source path) to a location specified in parameter 2 (target path).
Execute	Executes the application or script listed under parameter 1.

Source and target paths can contain virtual environment variables, which TwinCAT resolves accordingly.

The following environment variables are supported:

Virtual environment variable	Registry value	Default value
%TC_INSTALLPATH%	InstallDir	C:\TwinCAT\3.**x** \
%TC_TARGETPATH%	TargetDir	C:\TwinCAT\3.**x** \Target\
%TC_BOOTPRJPATH%	BootDir	C:\TwinCAT\3.**x** \Boot\
%TC_RESOURCEPATH%	ResourceDir	C:\TwinCAT\3.**x** \Target\Resource\
%SOLUTIONPATH%	-	Location of the solution file

Registry values are stored in the registry under the following key: *\HKLM\Software\Beckhoff\TwinCAT3.*

Example:

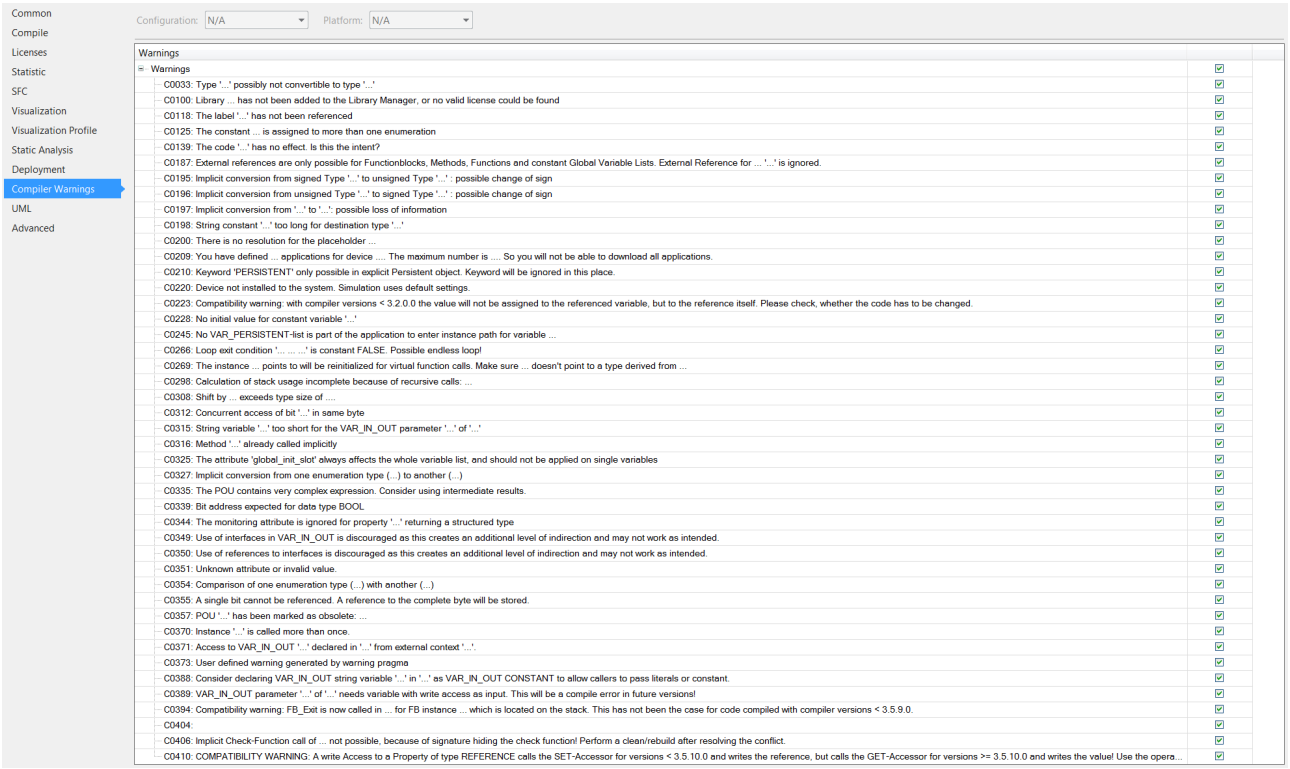
In the following example, the file *SampleFile.xml* file is copied from the Config project subfolder of the solution to the folder *C:\plc\config* on the target system.

Event	Command Type	Parameter1	Parameter2
Activate Configuration	Copy	%SOLUTIONPATH%Config\SampleFile.xml	C:\plc\Config\SampleFile.xml

5.4.4.10 Category Compiler Warnings

The **Compiler Warnings** category is used to select the compiler warnings that TwinCAT displays in the message window during a compilation run.

You can specify the maximum number of listed warnings in the **Compile** category.



See also:

- Command Build PLC project [▶ 115]
- Category Compile [▶ 97]

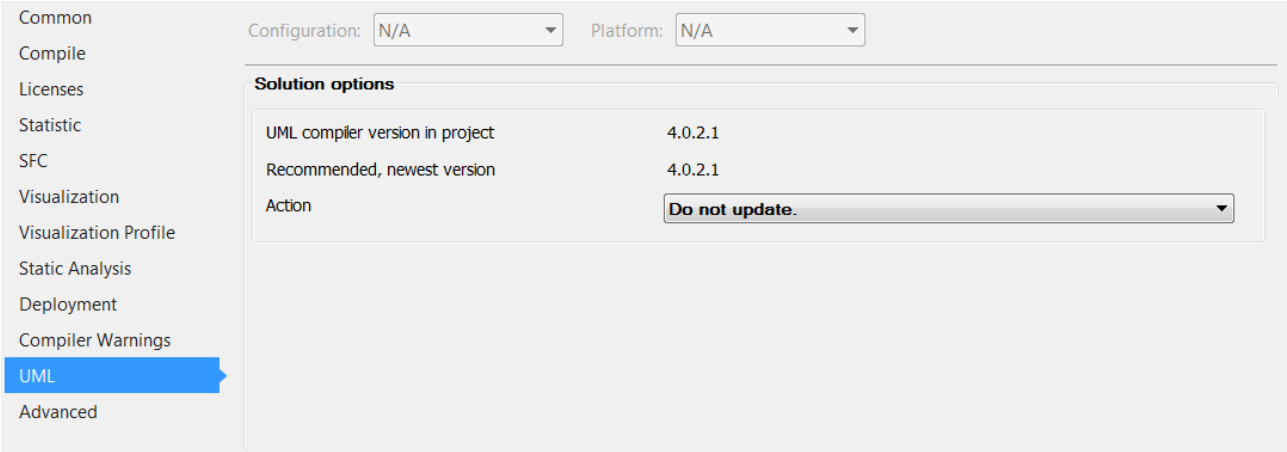
5.4.4.11 Category UML

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

In the category **UML** you can change the UML compiler version. This setting is only relevant when using the UML Statechart.

For more information on the configuration options, please refer to section "UML Compiler Version" of the TF1910 TC3 UML documentation.



5.4.4.12 Category Advanced

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **Advanced** is used to configure advanced properties.

Write options

Engineering incompatibility of file version 1.2.0.0 (or higher) with TC3.1 < build 4024

Please note that objects saved with file version 1.2.0.0 (or higher) cannot be loaded with Engineering versions lower than TC3.1.4024!

Since an object is automatically saved with file version 1.2.0.0 when using the optional Base64 format, objects with Base64 format cannot be loaded with Engineering versions lower than TC3.1.4024.

If a PLC project contains objects with the file version 1.1.0.1 and objects with the file version 1.2.0.0, the 1.1.0.1 objects are loaded with an Engineering version lower than TC3.1.4024. Objects with file version 1.2.0.0 are not loaded.

The file version of a file saved with file version 1.2.0.0 can be reset to 1.1.0.1 using XAE version TC3.1.4024 or higher.

Write object content as („Write object content as“)	Available from TC3.1 Build 4024 **Background information:** From build 4024, **Base64** introduces a new memory format that is optionally available for the following PLC objects: • POUs where the POU body is programmed in a graphical implementation language ◦ Sequential Function Chart (SFC) ◦ FBD/LD/IL (Function Block Diagram/Ladder Diagram/Instruction List) ◦ CFC (Continuous Function Chart and page-oriented CFC) ◦ UML class diagram and state diagram • POUs with a subelement (e.g., action, method) that is programmed in a graphical implementation language (for graphical languages see first key point) • Visualizations • Visualization Manager • Text lists • Recipe manager • Image pool Up to now, these objects were saved as XML by default. From build 4024 you can configure whether these object types should be saved as XML or Base64. **Advantages of Base64 over XML:** Base64 results in compressed storage, compared to XML. As a result, improved performance can be achieved with file access to these objects, which can be used, for example, when loading, moving or copying objects. **Setting option for the standard storage format:** For a PLC project, the setting "Write object content as" in the PLC project properties can be used to define the standard storage format for the object types mentioned above. The selected standard storage format is only used with newly added objects (exception: not with newly added POU sub-objects. Example: A POU is saved as an XML and the standard storage format is configured as Base64. If a graphic sub-object is then added to the POU, the storage format of the POU and thus of the sub-object as XML remains unchanged). The storage format of an existing object with a non-standard storage format is not automatically changed when the object is changed and saved. The storage format of an existing object can be changed individually via the Properties window (see below). Alternatively, when changing the standard storage format, there is the option of adopting the newly selected storage format for all existing objects. If you change the storage format at this point, a corresponding query window appears. The following options are available for the setting "Write object content as": • **XML** (default): The PLC objects mentioned above are saved in XML format by default. ◦ Objects with this storage format are stored in file version 1.1.0.1. • **Base64:** The PLC objects mentioned above are saved in Base64 format by default. ◦ Objects with this storage format are stored in file version 1.2.0.0. Please note that objects with the file version 1.2.0.0 (or higher) cannot be loaded with Engineering versions < TC3.1.4024! **Individual setting option of the storage format:** The storage format of an object can be configured individually for the object types mentioned above in the **Properties** window of the object. For more information see the description of the properties [▶ 90] (**Format** property).

Write product version in files („Write product version in files“)	Available from TC3.1 Build 4024 The product version indicates which plug-in version was used to save a PLC file (e.g., a function block). The setting of this checkbox is valid for the whole project and is the default setting for all modified or newly added PLC objects located in this PLC project.  (Default): The product or plugin version is written into the file (the version is not visible in XAE; it shows up if the file is analyzed at file level). • If you change the setting from disabled to enabled, a query window appears in which you can select whether to add the product version to all existing files. • Use case for the enabled option: This setting can be used to include the file version in the file for debugging or tracking purposes, for example. • Please note the following: If the file is saved with a different product version, this leads to a change of this file, which shows up as a file difference when using source code management systems.  : The product or plugin version is not written to the file. • If you change the setting from enabled to disabled, a query window appears in which you can select whether to remove the product version from all existing files. • Use case for the disabled option: This setting can be used if the product version is not of interest. This minimizes changes to files with regard to source code management systems.
Write object content with profile: (“Write object content with Profile”)	The profile defines the format in which objects are saved. With Build 4024, for example, new functionalities were added for the PLC HMI. For this reason, visualization files saved with Build 4024 cannot be directly opened with older builds. If you set a 4022 profile here, then the visualization files will be saved in the appropriate format and can be opened with Build 4022. **Requirement:** So that, for example, the 4022 profile is available in the drop-down menu, either a 4022 Remote Manager installation must be carried out or the current 4024 XAE installation must have been installed via a previously existing 4022 XAE installation.

Multiuser options

Use Multiuser („Use Multiuser“)	Available from TC3.1 Build 4024  (Default): The multiuser functionality of the PLC project is not enabled.  : The multiuser functionality of the PLC project is enabled. Please also refer to the further information in the multiuser documentation.

Solution options

Secure Online Mode („Secure Online Mode“)	☐ (Default): For security reasons, the user is always prompted to confirm the execution of the following commands when they are called. • Activate configuration • Restart TwinCAT System in Config/Run Mode • Reset cold • Reset origin ☑ : In addition to the above commands, for which a confirmation prompt appears by default, the following commands will also prompt you to confirm. • Start • Stop • Single Cycle
Autoupdate Visu Profile	This option allows you to configure the automatic update behavior of the visualization profile. When you open a PLC project that uses an outdated visualization profile, a warning appears in the message window ("New Version found for Visualization profile"). ☑ : In such a case, the visualization profile version is automatically set to the latest version if the option **Autoupdate Visu Profile** is enabled. With such an automatic update of the visualization profile version, a corresponding warning is displayed in the message window (e.g., "Visualization profile set from' TwinCAT 3.1 Build 4020.10' to' TwinCAT 3.1 Build 4022.0'"). ☐ (Default) If the **Autoupdate Visu Profile** option is disabled, the visualization profile version is not changed automatically. By double-clicking on the warning "New Version found for Visualization profile", you can open the ProfileUpdate dialog in which you can manually change the visualization profile version.
Autoupdate UML Profile	This option allows you to configure the automatic update behavior of the UML compiler version. If you open a PLC project, in which an outdated UML compiler version is used, a corresponding warning appears in the message window ("new version for UML found"). ☑ : In such a case, the UML compiler version is automatically set to the latest version if the option **Autoupdate Uml Profile** is enabled. In the case of such an automatic update of the UML compiler version, a corresponding warning will be displayed in the message window (e.g., "UML set from '4.0.2.0' to '4.0.2.1'"). ☐ (Default): If the option **Autoupdate UML Profile** is disabled, the UML compiler version is not changed automatically. Double-click on the warning “new version for UML found” to open the ProfileUpdate dialog, in which you can change the UML compiler version manually. For more information, see UML Compiler Version.

5.4.5 PLC project settings

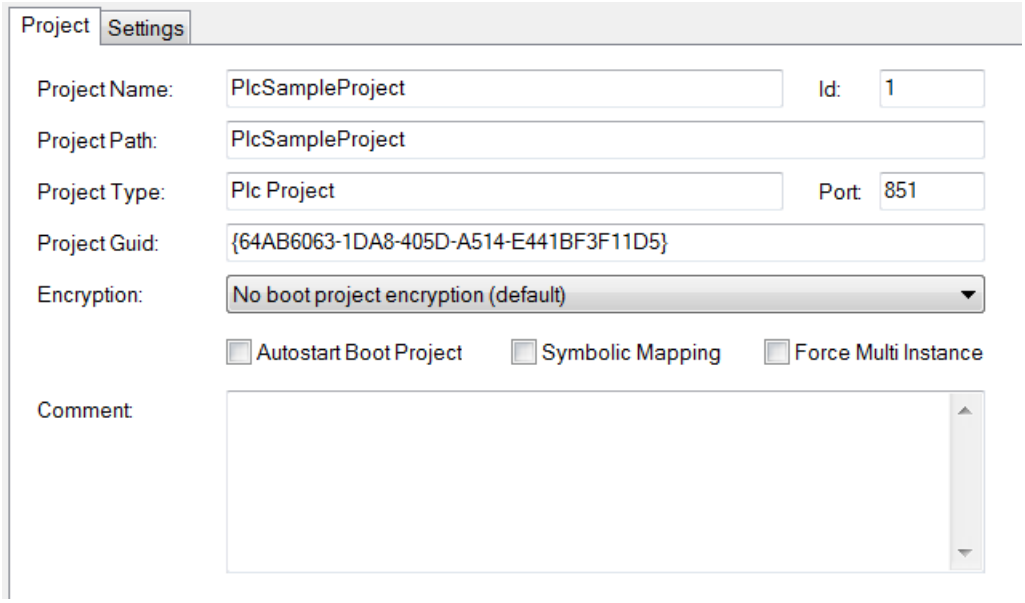
Function: This command opens an editor in which project settings can be defined.

Call: Double-click on the PLC project in the **Solution Explorer**

See also:

- PLC documentation: Configuring a project

5.4.5.1 Project tab

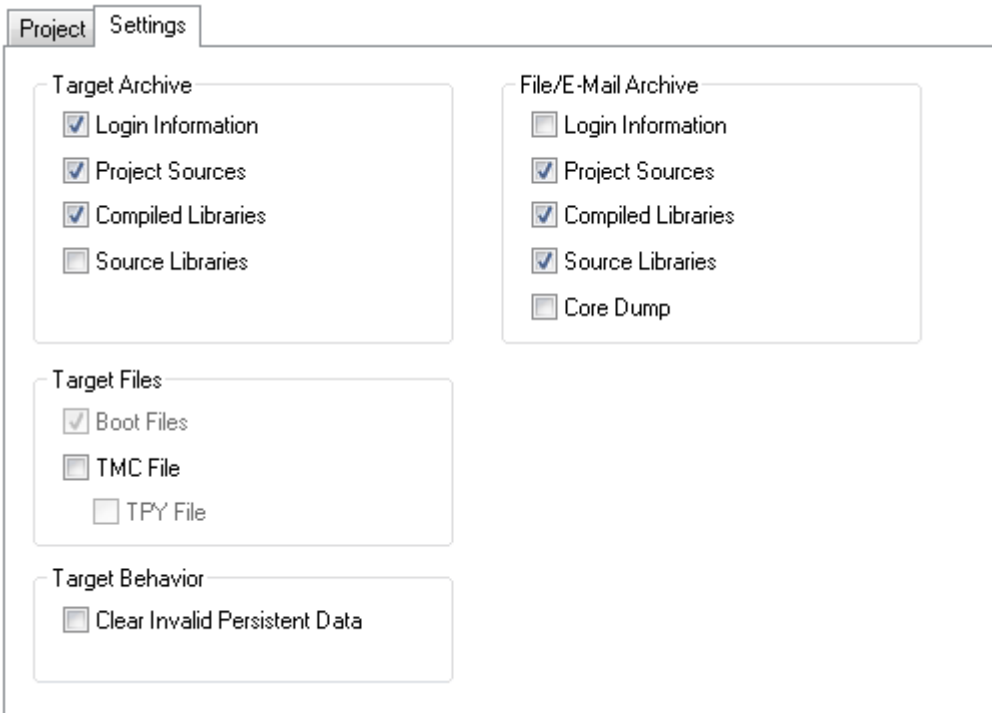


Project Name	Name of the PLC project
Id	ID of the PLC project
Project Path	Path to the location where the PLC project is stored
Project Type	Project type
Port	AMS port number of the runtime system
Project Guid	GUID of the PLC project
Encryption	Encryption of the boot project • No boot project encryption (default) • Encrypt boot project
Autostart Boot Project	☑ After the TwinCAT runtime environment has been started, the PLC boot project is automatically loaded and started. The setting is transferred directly to the currently selected target system. The setting is not saved in the TwinCAT project. This option corresponds to the **Autostart Boot Project** command in the context menu of the PLC project node in the Solution Explorer.
Symbolic Mapping	☑ Symbolic mapping is enabled.
Force Multi Instance	☑ Option for logging in multiple instances of the PLC project enabled.
Comment	Comment box
Compiler Defines (available from TC3.1 build 4024)	
Manual	Here you can define your own compiler definitions at System Manager level, which are passed on to the PLC project. The definitions are entered in the PLC project properties under the category compile as system compiler definitions [▶ 97].
Implicit	☑ The names of the selected project variants as well as all groups to which the project variant belongs are automatically set as a compiler definition and passed on to the PLC project. The definitions are entered in the PLC project properties under the category compile as system compiler definitions [▶ 97]. **Note:** In order to activate this checkbox, the Defines for the variant management must be enabled.

See also:

- Variant management: Concept: Integration in the PLC project
- Command properties (PLC project): Category Compile: System compiler definitions [▶ 97]

5.4.5.2 Settings tab



Target Archive

In the **Target Archive** group box you can specify which information is transferred to the target system together with other data when you create a boot project.

Login Information	COMPILEINFO file containing the compiler information of the PLC project.
Project Sources	Source code files of the PLC project in readable source code form.
Compiled Libraries	Libraries that are used in compiled form in the PLC project.
Source Libraries	Libraries that are used in legible source code form in the PLC project.

File/E-Mail Archive

In the **File/E-Mail Archive** group box, you can specify what information is stored when archiving a PLC project [▶ 52], a TwinCAT project [▶ 232] or a Solution [▶ 50]. If you activate the corresponding checkbox, the files described in the following table are stored in the project archive.

Login Information	COMPILEINFO file containing the compiler information of the PLC project.
Project Sources	Source code files of the PLC project in readable source code form.
Compiled Libraries	Libraries that are used in compiled form in the PLC project.
Source Libraries	Libraries that are used in legible source code form in the PLC project.
Core dump	Core dump file, which is located in the PLC project directory, and the compile info files, which are located in the "_CompileInfo" folder in the project directory. Note: The compile info files are also saved in the archive if the "Core Dump [▶ 135]" setting is activated, as these files are needed in order to be able to use the core dump.

Transfer of source code

If you have configured the target or file/email archive settings to include the project sources and/or source libraries in one of these archives, please note that the project sources and/or the source libraries (*.library) used in the project are contained in the ZIP archive in readable source code form when passing on/delivering the target system or when passing on the file/email archive.

Keep this in mind when configuring the settings described above and when storing and referencing libraries (*.library vs *.compiled-library).

For more information on library management, see section Using libraries.

Information about source code encryption can be found in the documentation on Security Management.

Target Files

In the **Target Files** group box you can set which information is transferred to the \Boot\Plc folder when you create a boot project on the target system.

TMC File	TMC file (TwinCAT Module Class) of a PLC project.
TPY File	TPY file (contains, among other information, project information, routing information, compiler information, target system information).

Target Behavior

Clear Invalid Persistent Data	The backup of the stored persistent data is ignored. This ensures that any invalid data is not accepted but instead discarded. See: Backup of persistent data [▶ 113]

Backup of persistent data

Persistent data is regularly stored in a .bootdata file in the TwinCAT\Boot folder during a TwinCAT system stop/shutdown. At the next system startup (TwinCAT Run mode) this file is read, and the persistent variables in the runtime system are initialized with the values from the file. The system renames the .bootdata file to .bootdata-old.

This backup file (.bootdata-old) of the persistent data is read at system startup if the file (.bootdata) containing the persistent data does not exist. This is an exception, but it can occur, for example, if an IPC without UPS experiences a power failure and TwinCAT could not shut down properly.

- If it is foreseeable that the contents of the backup file will not be usable at a new system start, you can enable the option **Clear Invalid Persistent Data** to ignore the backup file. This can be the case, for example, if batch information or tool data has be stored in a production facility and has to be up-to-date.
- If the structure of the persistent data (its data types or symbol paths in the program code) is changed due to online changes, it makes no sense to subsequently load an obsolete persistent data file. In this case, you should enable the **Clear Invalid Persistent Data** option in advance.

In both cases, you should also ensure that a current persistent data file is available. Function blocks such as FB_WritePersistentData (PLC Lib Tc2_Utilities) and UPS protection against sudden power failures are available for this purpose.

When using persistent data, the corresponding flags (BootDataLoaded and OldBootData) from the global structure PlcAppSystemInfo should always be evaluated (see documentation on System > Global Data Types).

If neither the regular file nor the backup file can be loaded or if they don't exist, the variables marked as PERSISTENT are reinitialized in the same way as other "normal" variables, either with their explicitly specified initial values or with the standard initializations.

See also:

- PLC documentation: Remanent Variables - RETAIN, PERSISTENT

5.5 Build

5.5.1 Command Build Solution

Symbol: 

Function: This command starts the compilation process or the code generation for all projects contained in the solution.

Call: **Build** menu or context menu of the solution

Requirement: The solution is selected.

All projects contained in a solution are compiled one after the other. This also concerns the projects (PLC, C++, etc.) integrated below a TwinCAT project. The steps performed for a PLC project are described in section Command Build PLC project [▶ 115].

5.5.2 Command Rebuild Solution

Function: The command starts the compilation process for all projects contained in a solution, even if it was previously compiled without errors.

Call: **Build** command or context menu of the solution

Requirement: The solution is selected.

When rebuilding the solution, it will first be cleaned (see also: Command Clean Solution [▶ 114] and subsequently built (see also: Command Build Solution [▶ 114]).

See also:

- Command Rebuild a PLC project [▶ 116]

5.5.3 Command Clean Solution

Function: This command starts the cleaning of all projects contained in the solution.

Call: **Build** menu or context menu of the solution

Requirement: The solution is selected.

All projects contained in the solution are cleaned in succession. This also concerns the projects (PLC, C++, etc.) integrated below a TwinCAT project. The steps executed for a PLC project are described in the section Command Clean PLC project [▶ 116].

5.5.4 Command Check all objects

Function: The command initiates a compilation run, i.e. a syntax check for all objects located in the project tree of the PLC project. This is primarily useful when creating libraries or when processing library projects.

Call: Context menu of the PLC project object (<PLC project name> Project) in the **Solution Explorer**

As opposed to the Command Build PLC project [▶ 115], in which only the objects used are checked, when this command is executed the syntax of all objects in the PLC project is checked.

The command does not lead to code generation. No file with information on the compilation run is created in the project directory.

5.5.5 Command Build TwinCAT project

Symbol: 

Function: This command starts the compilation process or the code generation for the currently active TwinCAT project.

Call: **Build** menu if a TwinCAT project is currently selected, or context menu of the TwinCAT project

Requirement: The TwinCAT project is selected.

All of the projects (PLC, C++, etc.) contained in the TwinCAT project are compiled one after the other. The steps performed for a PLC project are described in section Command Build PLC project [▶ 115].

See also:

- Command Rebuild a TwinCAT project [▶ 115]

5.5.6 Command Rebuild a TwinCAT project

Function: The command starts the compilation process or the code generation for the currently active TwinCAT project, even if it was last compiled without error.

Call: **Build** menu if a TwinCAT project is currently selected, or context menu of the TwinCAT project

Requirement: The TwinCAT project is selected.

When rebuilding the project, the TwinCAT project will first be cleaned (see also: Command Clean TwinCAT project [▶ 115] and subsequently built (see also: Command Build TwinCAT project [▶ 115]).

5.5.7 Command Clean TwinCAT project

Function: This command deletes the local compilation information for the currently active PLC project and updates the language model of all objects.

Call: **Build** menu if a TwinCAT project is currently selected, or context menu of the TwinCAT project

Requirement: The TwinCAT project is selected.

All of the projects (PLC, C++, etc.) contained in the TwinCAT project are cleaned one after the other. The steps executed for a PLC project are described in the section Command Clean PLC project [▶ 116].

See also:

- Command Rebuild a TwinCAT project [▶ 115]

5.5.8 Command Build PLC project

Symbol: 

Function: This command starts the compilation process or the code generation for the currently active PLC project.

Call: **Build** menu if a PLC project is currently selected, or context menu of the PLC project object (<PLC project name> Project) in the **Solution Explorer**

Requirement: The PLC project is selected.

During the compilation, TwinCAT carries out a syntactic test of all objects used in the PLC project. The compilation procedure is always carried out automatically when you wish to log the project in with a changed program. After the check has been completed, TwinCAT displays any error messages or warnings in the Error List [▶ 83] view.

Apart from that, the compilation information of the PLC project is created when building the project and saved in a local file (*.compileinfo) in the Solution.

If the program was not changed since the last error-free compilation process, it is not recompiled. If the syntactic test is to be performed anyway, use the Command Rebuild a PLC project [▶ 116].

5.5.9 Command Rebuild a PLC project

Function: The command starts the compilation process or the code generation for the currently active PLC project, even if it was last compiled without error.

Call: **Build** menu if a PLC project is currently selected, or context menu of the PLC project object (<PLC project name> Project) in the **Solution Explorer**

Requirement: The PLC project is selected.

When rebuilding the project, the project will first be cleaned (see also: Command Clean PLC project [▶ 116] and subsequently built (see also: Command Build PLC project [▶ 115]).

5.5.10 Command Clean PLC project

Function: This command deletes the local compilation information for the currently active PLC project and updates the language model of all objects.

Call: **Build** menu if a PLC project is currently selected, or context menu of the PLC project object (<PLC project name> Project) in the **Solution Explorer**

Requirement: The PLC project is selected.

The compilation information was created during the last building of the project or the last online change or download of the PLC project and was saved in a local file (*.compileinfo) in the project folder. Unless configured otherwise (see also: PLC project settings, Settings tab [▶ 112]), the compilation information is transmitted to the target system during an online change or download. When the PLC project is cleaned, only the local compilation information is removed. The compilation information on the target system is retained.

See also:

- Command Rebuild a PLC project [▶ 116]

5.6 Debug

5.6.1 Command New Breakpoint

Symbol: 

Function: The command opens the **Breakpoint Properties** dialog.

Call: Debug menu, button **New** in the **Breakpoint** view (**PLC > Window > Breakpoints**).

Requirement: The PLC project is in online mode.

The command **Toggle Breakpoint** can be used to set a new breakpoint directly at the current cursor position in online mode.

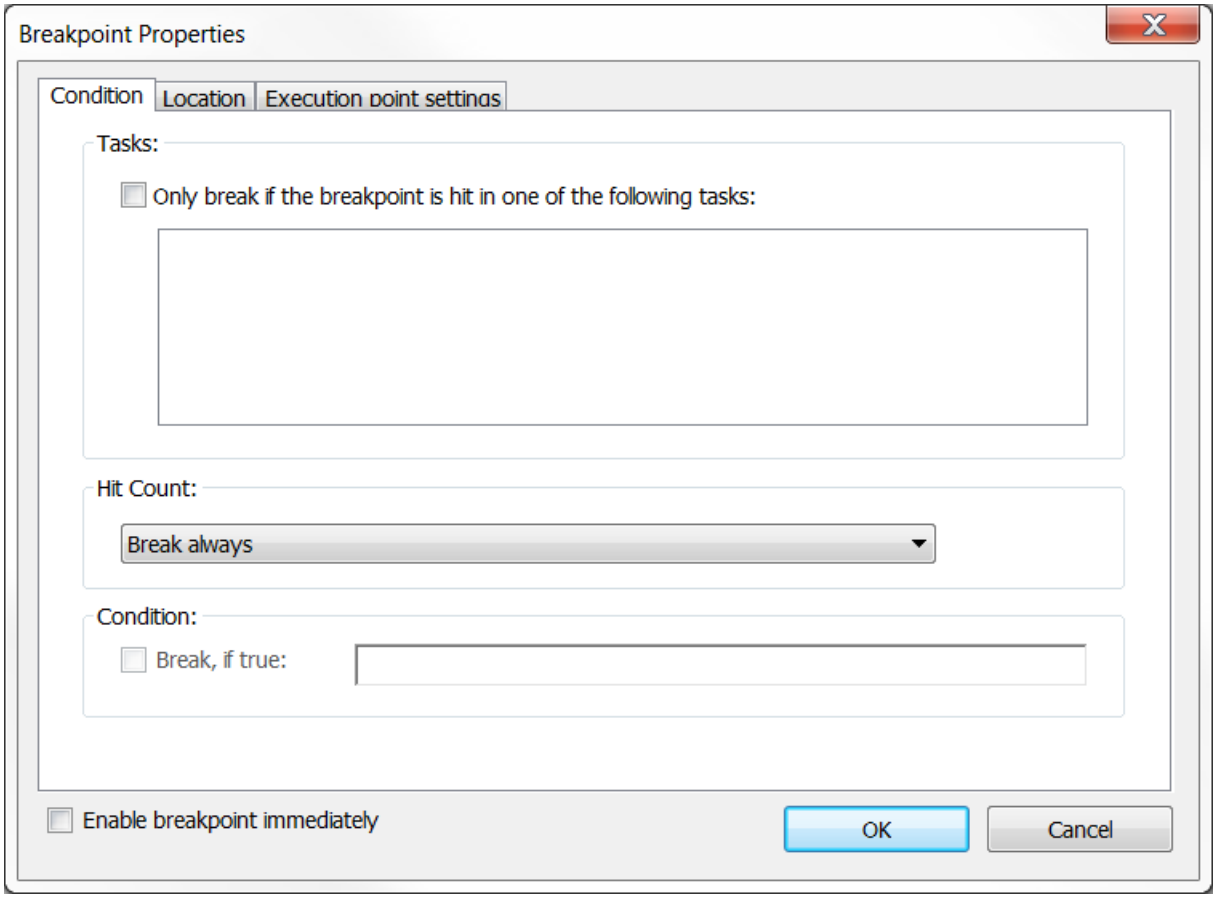
See also:

- Command Toggle Breakpoint [▶ 120]
- PLC menu: Command Breakpoints [▶ 128]
- PLC documentation: Use of breakpoints

Breakpoint Properties dialog

Condition tab

The dialog defines the conditions under which the program execution should stop at the breakpoint.



Tasks

Only break if the breakpoint is hit in one of the following tasks	☑: TwinCAT only evaluates the breakpoint if it is reached by certain tasks. The required tasks must be activated. For example, you can define a single "debug task" and thus prevent other tasks that also use the function block from being affected during debugging.

Hit Count

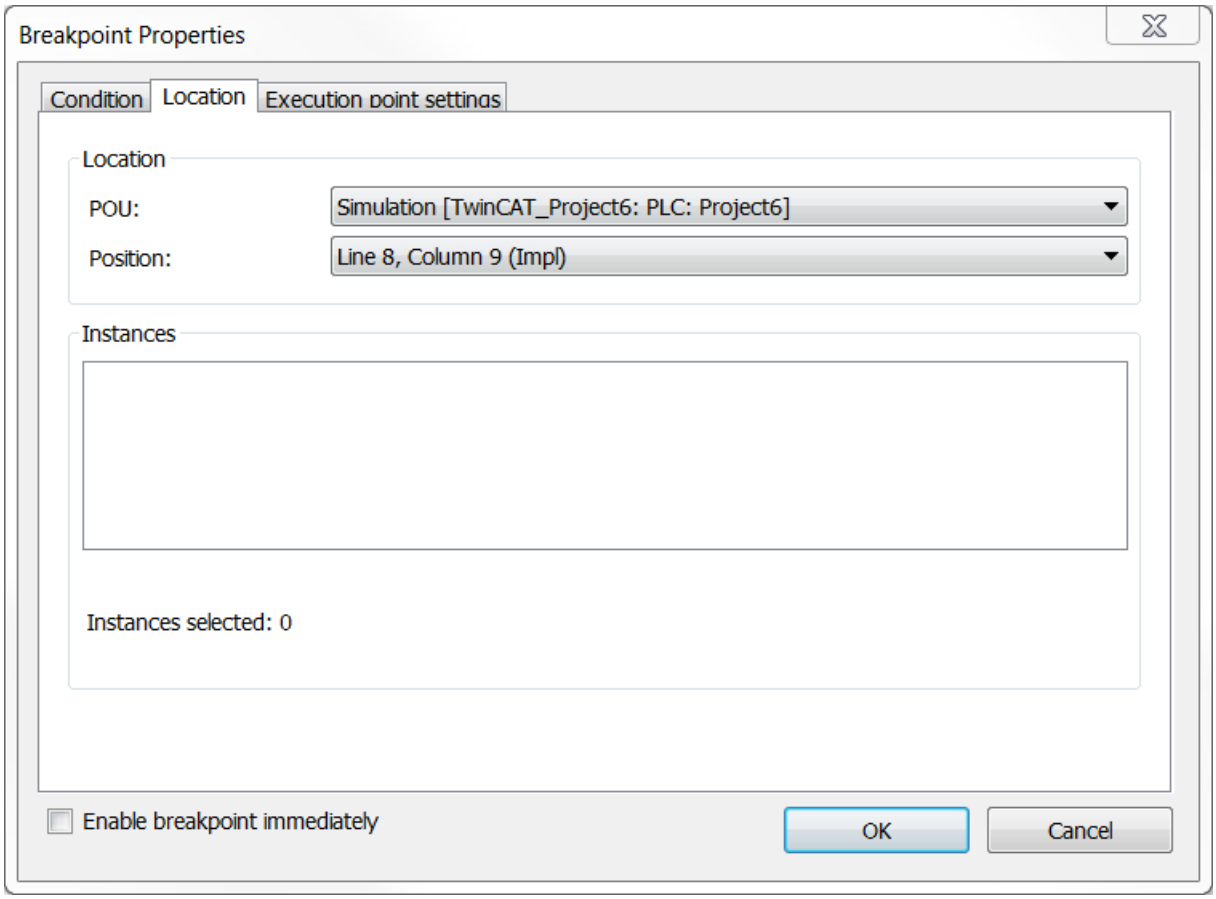
Hit Count	Break always: The program always stops at this breakpoint. Alternative: The program stops at the breakpoint when the breakpoint is hit as often as defined below (enter the desired number of hits or select from the number list): • Break if the Hit Count matches • Break if the Hit Count is a multiple of • Break if the Hit Count is greater or equal

Condition

Break, if TRUE	Definition of conditional breakpoints. The condition can only be entered in online mode.  : TwinCAT evaluates the specified condition and stops the program at this breakpoint if the result is TRUE. Valid Boolean expressions can be entered as a condition. Examples: x>100, x[y]=z, a AND b, boolVar.

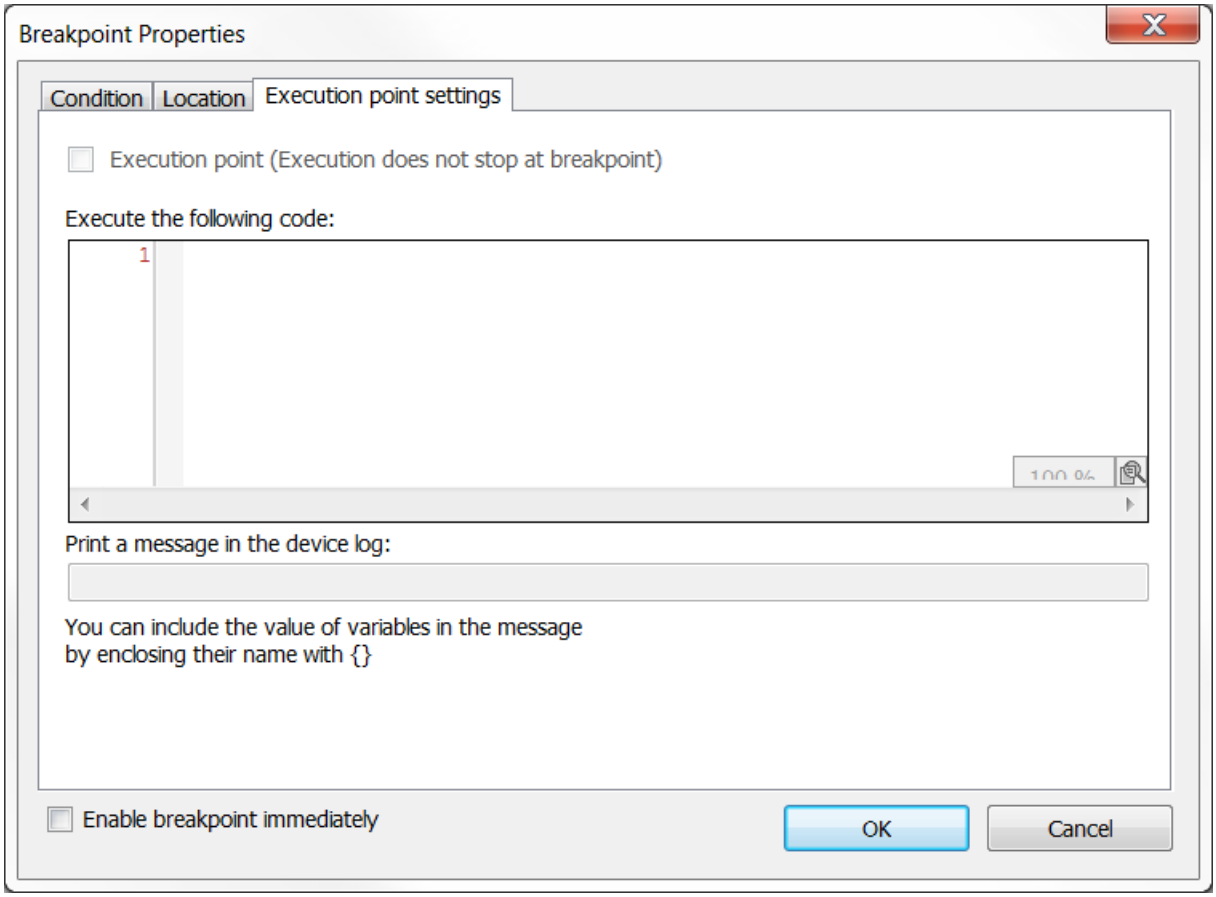
The use of conditional breakpoints slows the code execution, even if the condition is not TRUE.

Location tab



POU	Function block of the active PLC project in which the breakpoint is to be positioned.
Position	Position of the breakpoint in the POU. Specification in the form of line and column numbers (text editor) or as network or item numbers.
Instance path:	For function blocks, you must specify whether the breakpoint should be set in the implementation or in an instance ☑ TwinCAT sets the breakpoint in the instance. With this option, you select the instance path. ☐ TwinCAT sets the breakpoint in the implementation.

Execution point settings tab



Execution point (Execution does not stop at breakpoint)	☑: The breakpoint becomes the execution point. The execution does not stop at this point, but the specified code is executed. activated: ●, deactivated: ○
Execute the following code	Code that is executed when the execution point is reached. Loop constructs (For, While) and IF or CASE expressions are not possible.
Print a message in the device log	This option is not available.

See also:

- PLC documentation: Use of breakpoints

5.6.2 Command Edit Breakpoint

Symbol: 

Function: The command opens the **Breakpoint Properties** dialog.

Call: Debug menu, button  in the **Breakpoints** view (**PLC > Window > Breakpoints**)

Requirement: The PLC project is in online mode. The cursor is on a breakpoint.

See also:

- Command New Breakpoint > Breakpoint Properties dialog [▶ 117]
- PLC documentation: Use of breakpoints

5.6.3 Command Enable Breakpoint

Symbol: 

Function: The command enables a disabled breakpoint.

Call: Debug menu, button  in the **Breakpoints** view (**PLC > Window > Breakpoints**)

Requirement: The PLC project is in online mode. The cursor is on a disabled breakpoint.

See also:

- PLC documentation: Use of breakpoints

5.6.4 Command Disable Breakpoint

Symbol: 

Function: The command disables an enabled breakpoint.

Call: Debug menu, button  in the **Breakpoints** view (**PLC > Window > Breakpoints**)

Requirement: The PLC project is in online mode. The cursor is on an enabled breakpoint.

See also:

- PLC documentation: Use of breakpoints

5.6.5 Command Toggle Breakpoint

Hotkey: **[F9]**

Function: The command sets a breakpoint or removes an existing breakpoint.

Call: Debug menu

Requirement: The PLC project is in online mode. The cursor is on a breakpoint.

See also:

- PLC documentation: Use of breakpoints

5.6.6 Command Step over

Symbol: 

Hotkey: **[F10]**

Function: The command executes a program in defined steps.

Call: Debug menu, **TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode. The program is at a program step.

In a Step Into statement, the command corresponds to the **single step**. However, if the processing reaches a function block call, the **step over** results in complete processing of the called function block within the current step. A complete action is executed in a sequence chart.

Use the **Step Into** command to execute a called function block step-by-step.

See also:

- Command Step into [▶ 121]
- PLC documentation: Stepping

5.6.7 Command Step into

Symbol: 

Hotkey: **[F11]**

Function: The command executes a program in single steps.

Call: Debug menu, **TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode. The program is at a program step.

In single step mode, the program stops before the next statement. If necessary, the execution switches to an another POU. If the current position is a function call or function block call, the execution stops before the first statement of the called function block.

In all other situations, the command has the same effect as the command **Step Over**.

See also:

- Command Step over [▶ 121]
- PLC documentation: Stepwise processing of the program (stepping)

5.6.8 Command Step out

Symbol: 

Hotkey: **[Shift] + [F11]**

Function: The command executes the program up to the program start, or the calling program.

Call: Debug menu, **TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode. The program is at a breakpoint.

The command executes the program up to the program start, or the calling program.

If a program does not contain any calls, the command causes the program to run until it is started. If the current position is within a called POU, this command results in execution up to return to the calling function block. In the case of nested calls, you can therefore use this command to go back step-by-step in the hierarchy of calls.

See also:

- PLC documentation: Stepwise processing of the program (stepping)

5.6.9 Command Show Next Statement

Symbol: 

Function: This command jumps to the program statement that will be executed in the next step.

Call: Debug menu

Requirement: The PLC project is in online mode. The program is at a program step.

This is useful when you have placed the cursor elsewhere or at a different POU during step-by-step processing of a program. The window of the corresponding function block reappears in the foreground, and the cursor is placed before the next statement to be executed.

See also:

- PLC documentation: Stepwise processing of the program (stepping)

5.6.10 Command Set next statement

Symbol: 

Function: The command determines which statement is executed next.

Call: Context menu

Requirement: The PLC project is in online mode. The program is at a program step.

Place the cursor at the desired statement and select the command. In this case the statements between the current statement and the selected statement are not executed.

See also:

- PLC documentation: Stepwise processing of the program (stepping)

5.6.11 Command Run To Cursor

Symbol: 

Function: The command executes a program up to a freely definable position.

Call: Context menu

Requirement: The PLC project is in online mode. The program is at a program step.

Place the cursor at the desired stop position and select the command. The statements that lie between the current and the defined position are then executed in one step.

See also:

- PLC documentation: Stepwise processing of the program (stepping)

5.7 TwinCAT

5.7.1 Command Activate configuration

Symbol: 

Function: The command enables a new configuration. The previous old configuration will be overwritten.

Call: Menu **TwinCAT**, **TwinCAT XAE Base toolbar options**

Within the confirmation window that appears after this command is executed, you can set whether the **Autostart boot project** setting should be activated for all PLC projects in the TwinCAT project.

See also:

- Command Activate boot project [▶ 236]
- Command Autostart boot project [▶ 236]

5.7.2 Command Restart TwinCAT System

Symbol: 

Function: The command starts TwinCAT in Run mode.

Call: Menu **TwinCAT**, **TwinCAT XAE Base Toolbar Options**

5.7.3 Command Restart TwinCAT (Config mode)

Symbol: 

Function: The command starts TwinCAT in configuration mode (Config mode).

Call: Menu **TwinCAT**, **TwinCAT XAE Base Toolbar Options**

5.7.4 Command Reload Devices

Symbol: 

Function: The command loads the created I/O devices.

Call: Menu **TwinCAT**, **TwinCAT XAE Base Toolbar Options**

5.7.5 Command Scan

Symbol: 

Function: The command starts a device scan. The system searches for available I/O devices, connected "boxes" and, if applicable, bus modules and IP-Link extension modules.

Call: Menu **TwinCAT**, **TwinCAT XAE Base Toolbar Options**

Requirement: The "I/O" object is selected in the TwinCAT project structure in the **Solution Explorer**.

5.7.6 Command Toggle Free Run State

Symbol: 

Function: The command sets found I/O devices to free-run mode. This means, for example, that Bus Terminals can set (write) I/O channels to a certain status, without a PLC project or another triggering task being active.

Call: Menu **TwinCAT**, **TwinCAT XAE Base Toolbar Options**

Requirement: The system is currently in configuration mode.

If the target system was previously in Run mode, the command **Reload Devices** must be executed once before the I/O drivers for the device can be set to free-run state.

5.7.7 Command Show Online Data

Symbol: 

Function: The command connects to the selected target system and displays the parameter values and settings that are active on the target system in the corresponding views.

Call: Menu **TwinCAT**, **TwinCAT XAE Base Toolbar Options**

5.7.8 Command Choose Target System

Function: Drop-down list to select the target device for the control application.

Call: TwinCAT XAE Base Toolbar Options

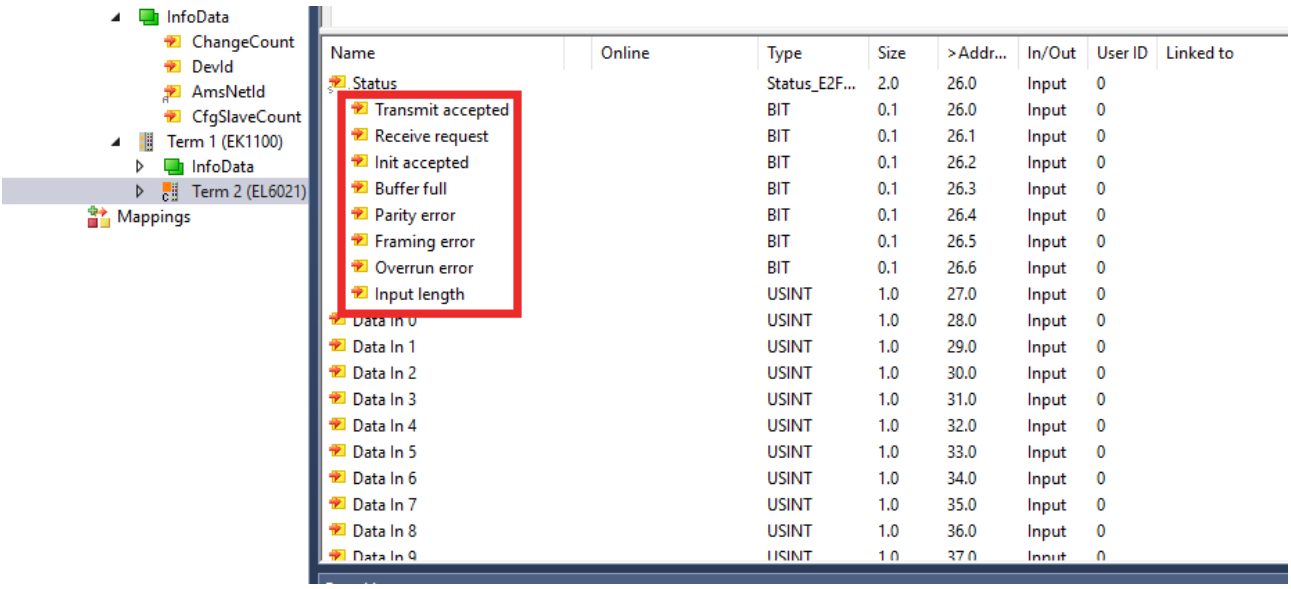
Select <Local> to load the control code directly into the local runtime of your programming device. If you want to select another target device, select **Choose Target System** from the drop-down list.

5.7.9 Command Show Sub Items

Symbol: 

Function: The command displays the subelements of an element with their properties and values in the overview view of a device. The command can be enabled or disabled. The command does not refer to the representation of the elements in the TwinCAT project tree.

Call: Menu **TwinCAT, TwinCAT Base XAE toolbar options**



5.7.10 Command Software Protection

Symbol:

Function: The command opens the **Software Protection** dialog.

Call: **TwinCAT** menu

In the Software Protection dialog you can define the security and user settings for a TwinCAT project.

Further information about the security and user settings can be found in the Software Protection documentation.

5.7.11 Command Hide Disabled Items

Symbol:

Function: This command makes it possible to make disabled objects invisible and visible again in the entire project tree. In this way, the display can be limited to active objects, thereby increasing clarity within the project tree.

Call: Menu **TwinCAT**, **TwinCAT XAE Base toolbar options**

5.8 PLC

5.8.1 Window

5.8.1.1 Command Watch List <n>

Symbol:

Function: The command opens the view **Watch List <n>**. You can fill a Watch List with variables from your project in order to be able to monitor, force or write the values for these variables in online mode within a single view. n can be 1, 2, 3, 4, which means that you can configure up to four Watch Lists.

Call: Menu **PLC > Window**

See also:

- PLC documentation: Using Watch Lists

5.8.1.2 Command Watch all forces

Symbol: 

Function: The command opens the **Watch all forces** view, which is a special form of a Watch List.

Call: Menu **PLC > Window**

Requirement: A PLC project is open in offline or online mode.

The view contains all variables of the PLC project currently prepared for forcing and all forced variables of the PLC project in a list. In the list you can perform the actions that are also possible in other Watch Lists. In addition, the **Unforce** selection menu contains the following commands:

- **Unforce and keep all selected values:** For all selected entries in the list, the variables are set to the forced value and forcing is canceled.
- **Unforce and restore all selected values:** For all selected entries in the list, the variables are reset to the value they had before forcing, and forcing is canceled.

See also:

- PLC documentation: Forcing and Writing Variables Values
- PLC documentation: Using Watch Lists

5.8.1.3 Command Cross Reference List

Symbol: 

Function: The command opens the **Cross Reference List** view.

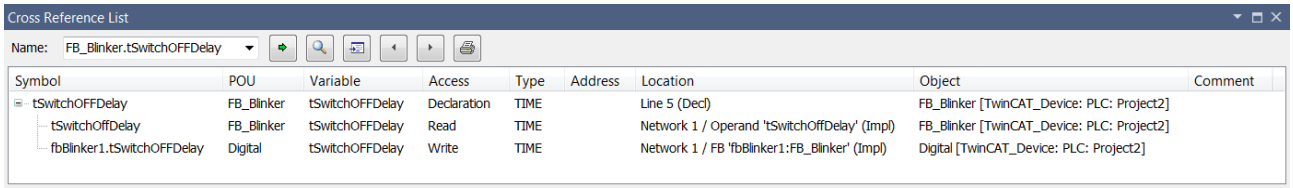
Call: Menu **PLC > Window**

Cross Reference List view

This view shows a list of cross-references for a symbol in the project. The symbol can be a variable, a POU (program, function block, function), or a user-specific data type (DUT).

The cross-reference list offer two basic types of searches:

- Text search: By specifying a symbol name, the cross-references of all symbols in the project are displayed with their names. If multiple symbols with the same name are found, then the display can be limited to individual declarations by means of the context menu.
- Declaration search: The symbol can be selected by means of the input assistant or by specifying a qualified path, for example MAIN.nVar. Then only the occurrence locations of this symbol are displayed, even if there exist other symbols with the same name.



Toolbar

Name	Symbol name (variable name, POU name, DUT name). Input options: • Selection of a declared symbol by means of the input assistant ( button) • Manual input of the symbol name. Triggering of the search by pressing the  button or the [Enter] key. For the text search, you can use the placeholders “*” (any number of characters) or “?” (exactly any one character) in combination with a partial string of a variable identifier. Use the percent sign “%” to search for IEC addresses. Examples: “%MW8”, “%M*”. Additional possibilities from outside the **Cross Reference List** view: • Use the **Find All References** command in the context menu if the name of a declared symbol is selected in an editor, or if the cursor is in the name field. • Automatic if the name of a declared symbol is selected in an editor, or if the cursor is in the name field. A automatic search is also possible if the object is selected in the project tree. Requirement: the **Cross Reference List** view is opened and the TwinCAT option **Automatically list selection in cross reference view** (category **Smart Coding**) is activated. The following input is valid: • Variable name, simple or fully qualified. Examples: “nVar”, “MAIN.nVar”. • POU name: Examples: “MAIN”, “FB_MyFB”. • DUT name: Example: “ST_MySTRUCT”” • Strings combined with placeholders: “*” (for any character) or “?” (for exactly one character). Example: “nVar*” applies to nVar1, nVarGlob2, nVar45 etc.... “nVar?” refers to nVar1, nVar2, nVarX etc., but not nVarGlob2, nVar45 etc. • “%<IEC address>”: TwinCAT searches for variables that are assigned to this address and direct memory access. Example “%QB0”, “%Q0 := 2”. Upper/lower case and spaces at the beginning and end of the input string are ignored.
	Find cross-references: The search is performed.
	Open input assistant for selecting a symbol.
	Show source position of selected cross-reference: The focus jumps to the occurrence location of the symbol.
	Show the source position of the previous cross-reference
	Show the source position of the next cross-reference
	Print Cross Reference List: The standard dialog for setting up a print job appears.

Table of cross-references found

Symbol	The locations for the symbols (variables, POUs, DUTs) are grouped by declaration. The declaration occurrence comprises the root node and the occurrence locations in the project are indented below. The precise expression is displayed that has the symbol at the occurrence location. Example: If there is a global variable “nVar” in the project and a local declared variable “nVar” in a POU, then two root node entries will be listed after a text search for cross-references with the occurrences of the variable “nVar” below each.
POU	POU name, DUT name; also task name in case of a function block call in the task configuration, for example.
Variable	Pure variable name, for example “nVar”.
Access	Type of access to the variable at the occurrence location: Declaration / Read / Write / Call. Special case for pointers: An assignment of type `pSample := ADR(nVar1)` is displayed as Write | Address when searching for “nVar1”. Reason: Any write access to pSample is not displayed when searching for “nVar1”. Write access is also possible by means of pointer variables.
Type	Data type of the variable.
Address	IEC address, if assigned to the variable. Example: “AT%QB0”.
Position	Location of the occurrence in the POU editor, for example line number, network number, declaration part, or implementation part. Example: “Line 1, Column 1 (Impl)” or “Line 9 (Decl)).
Object	POU name plus complete path of the occurrence location in brackets. Example: "MAIN [TwinCAT_SampleProject: PLC: SamplePLCProject]"
Comment	Comments if available in the declaration of the variable

The search returns all locations in the project as well as in inserted, uncompiled libraries.

Commands in the context menu of the Cross Reference List

Show source position	Opens the respective POU and marks the occurrence: for root entries, the declaration, and for subordinate entries, the respective occurrence location. As an alternative, you can double-click a line.
Limit results to current declaration	Limits the display of results to the selected symbol declaration if multiple declarations are found.
Expand All	In the list, every single result location is shown.
Collapse All	In the list, only the root nodes of the result locations are shown.

See also:

- Command 'Limit results to current declaration'
- Command Collapse All Folds
- Command Expand All Folds
- PLC documentation: Using the Cross Reference List to find Occurrences

5.8.1.4 Command Breakpoints

Symbol: 

Function: The command opens the **Breakpoints** view.

Call: Menu **PLC > Window**

Breakpoints view

The view provides an overview of all defined breakpoints of an application. All breakpoint commands are available within the view.

Table of current breakpoints

Application	Select the desired PLC project from the list.
POU	Name of the function block containing the breakpoint.
Position	Breakpoint position within the POU • Text Editor: Row and column number • Graphic editor: Network or element number "(Impl)" in the case of function blocks indicates that the breakpoint is in the implementation of the function block, not in an instance.
Instance path	Complete object path of the breakpoint position.
Tasks	Name of the tasks for whose execution the breakpoint is to be effective. If there is no restriction, it says "(all)".
Condition	• Break always: No additional activation condition defined. The breakpoint is always active. • Boolean expression. The expression must return TRUE for the breakpoint to be active.
Hit count condition	Indicates when (in which dependency on the hit count) the breakpoint should become effective.
Current hit count	Specifies how often the breakpoint has already been passed through ("hit") during execution.

Toolbar

	New Breakpoint (corresponds to the command Command New Breakpoint [▶ 116] in the **Debug** menu)	Opens the **Breakpoint Properties** dialog
	Clear breakpoint	Removes the breakpoint. Do not confuse this command with the Disable command.
	Enable/disable breakpoint (corresponds to the commands Command Enable Breakpoint [▶ 120] and Command Disable Breakpoint [▶ 120] in the **Debug** menu)	Switches the breakpoint or execution point between "enabled" and "disabled" status. • breakpoint enabled • breakpoint disabled • Execution point enabled • Execution point disabled In contrast to **Clear breakpoint**, a disabled breakpoint remains in the list and can be enabled again.
	Properties	Opens the **Breakpoint Properties** dialog for editing the breakpoint parameters. In online mode, you can convert the breakpoint to a execution point here.
	Go to source code position	Opens the online view of the respective function block. The cursor is at the breakpoint position.
	Delete All Breakpoints	Deletes all breakpoints and execution points of the application. The list is emptied. Not to be confused with disabling!
	Enable all breakpoints	Enables all currently disabled breakpoints and execution points.
	Disable all breakpoints	Disables all currently enabled breakpoints and execution points. The points remain in the list and can be re-enabled.

See also:

- Command New Breakpoint > Breakpoint Properties dialog [▶ 117]
- Command Toggle Breakpoint [▶ 120]
- PLC documentation: Use of breakpoints

5.8.1.5 Command Call Stack

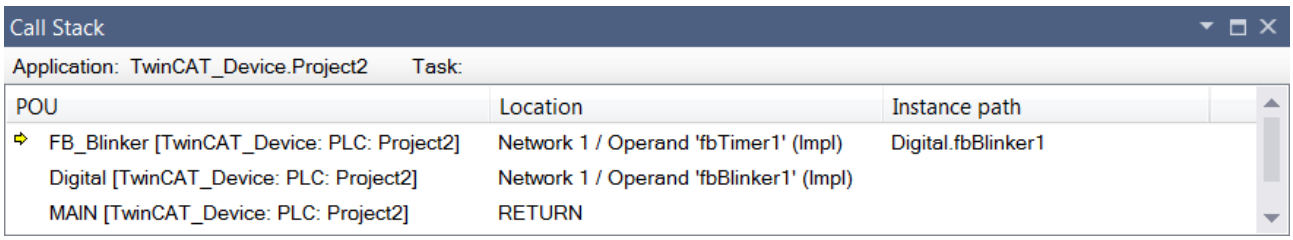
Symbol: 

Function: The command opens the **Call Stack** view.

Call: Menu **PLC > Window**

Callstack view

This view is useful if you want to run programs step-by-step. It shows the currently reached position with the complete call path.



Application	Name of the active PLC project that controls the currently reached program block.
Task	Name of the task that controls the currently reached program block.

POU	Name of the program block in which the program execution is located. The first line in the list describes the current execution position. It is marked with a yellow arrow. If this position is in a function block called by another function block, the position of the call is described in the second line. If the caller is called by another function block, this call position is described in the third line and so on.
Location	Position within the program block where the program execution is located • Row and column number for text editors • Network or element number for graphical editors
Instance path	Instance in which the program is executed.

The Call Stack is also available in offline mode and in normal online mode if you are not using any debugging functions. In this case, it contains the last position displayed during a step-by-step execution, but in "gray" font.

In contrast to the **Call Stack** view, the **Call Tree** view provides call information on a function block at any time.

See also:

- PLC documentation: Use of breakpoints

5.8.1.6 Command Memory

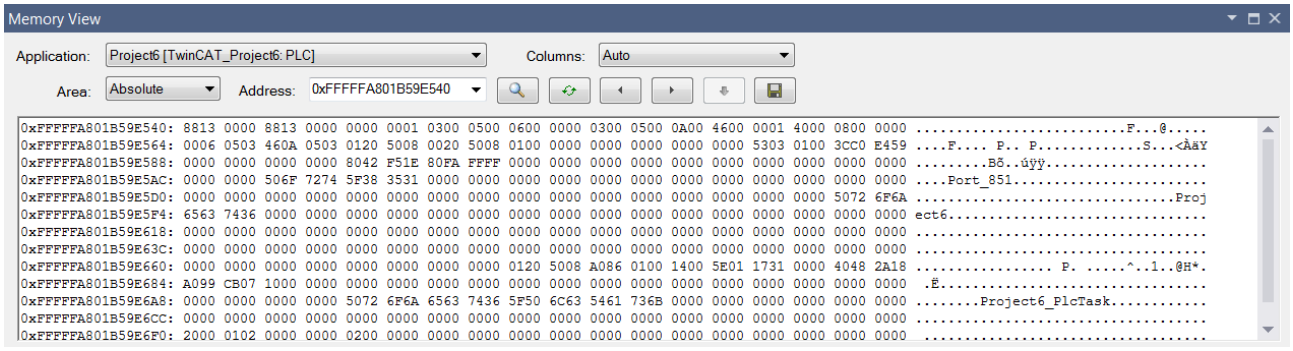
Symbol: 

Function: The command opens the **Memory View**.

Call: Menu **PLC > Memory**

Requirement: As a rule, the control system supports the functionality. At least one application is loaded and in online mode.

Memory View



Application	Select the PLC project for which the memory view is to be displayed. You must be logged on to the controller with this project. It does not have to be the "active PLC project".
Area	• Absolute: Memory is addressed directly and completely. The address is in the input field next to it. • Area <i>: Memory areas of the controller, starting with Area 0. Memory areas reserved exclusively for code are not displayed.
Address	Absolute start address of the core dump Requirement: Absolute is selected in the range.
Offset	Address offset to the selected memory area in bytes, for example 0x0200, 16#0200 or as decimal number 512 Requirement: A memory area is selected in area, e.g. Area 0. TwinCAT offers all currently used memory areas for selection. Memory areas reserved exclusively for code are not displayed.
	Find out the address for a variable: Input Assistant for selecting an IEC variable appears. If you have selected a variable, TwinCAT presets the start address with the variable address.
	Load/update memory view
	Show previous segment: Navigate to the previous memory segment
	Show next segment: Navigate to the next memory segment
	**Note TwinCAT does not check whether the changes are permissible. You can crash the application by making inadvertent changes** Load changes to PLC: TwinCAT transfers the new data to the controller. Requirement: You have overwritten one or more bytes in the memory view.
	Save the contents of the memory to a file: The dialog "Memory content as binary file" appears. Select a location.
Columns	Number of columns in the hexadecimal representation of the memory dump. Two bytes are displayed per column. With Auto, the number of columns adjusts to the window size. To the right of it, the data is displayed as text.

5.8.1.7 Command Call Tree

Symbol: 

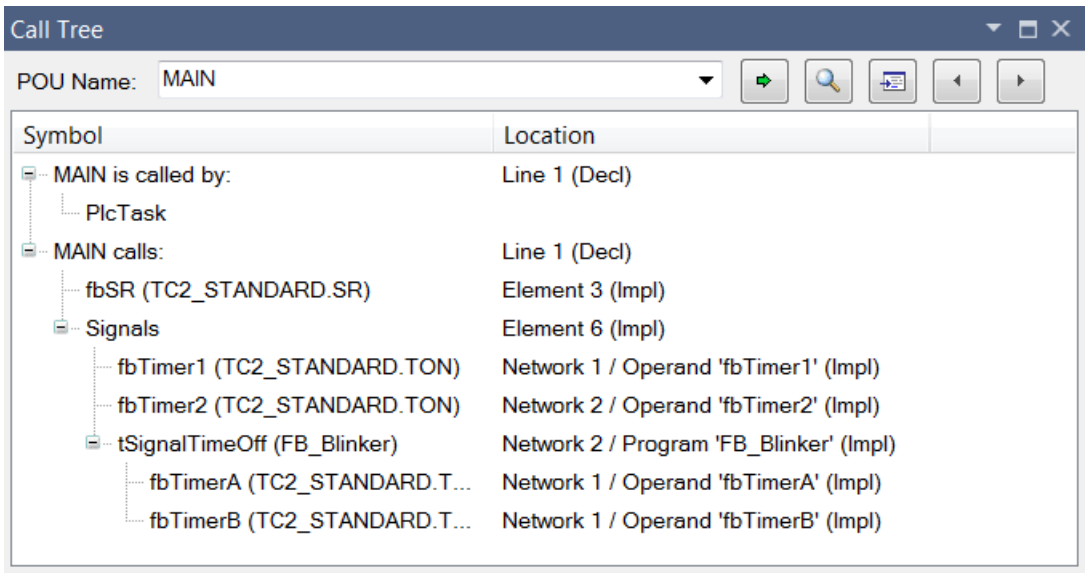
Function: The command opens the **Call Tree** view.

Call: Menu **PLC > Window**

Call Tree view

The Call Tree is available at any time before the application is compiled. It is a static representation of the callers and calls of the function block, which you can specify explicitly. This means that the tree always contains two root nodes, under which the respective call sequence is displayed as consecutive indented entries. Recursive calls are quickly recognizable in this tree view.

Example of a Call Tree (1) for function block (2) MAIN:



- (3) Node "<function block name> is called by:"
- (4) Node "<function block name> calls:"

POU Name	The name of the program block can be entered manually, by dragging it from another view, or by using the button. The selection list contains the last function block names entered.

Toolbar and keyboard operation

< Find function block	TwinCAT searches for the function block specified in "Function block name" and displays its callers and calls.
Take function block from Input Assistant	The **Input Assistant** dialog appears for selecting a function block call or instance call. The Call Tree is automatically updated after the selection.
Display source code position of the selected function block	TwinCAT jumps to the point where the function block is used in the source code of your program.
Show source code position of the next function block Show source code position of the previous function block	The selection in the Call Tree jumps to the next or previous function block in the call structure. At the same time, the corresponding source code position is opened in the respective editor. Double-clicking on an entry in the Call Tree also opens the corresponding source code position.

Display of the Call Tree

Location	For the root nodes in the Call Tree: Line number of the declaration ("Decl") of the function block. For the callers or calls under the root node: Depending on the implementation language, line number, column number, network number of your position.

Context menu for the entry currently selected in the tree

Collapse All	The expanded entries in the Call Tree are collapsed except for the two root nodes.
Display source code position	TwinCAT jumps to the point where the function block is used in the source code of your program.
Set as new root node	The entry selected in the Call Tree appears in "function block name". The tree is automatically adapted for the new root nodes.

In contrast to the static Call Tree, which always provides call information for a function block, the **Call Stack** view is intended for immediate information during the step-by-step processing of a program. The Call Stack always shows the complete call path of the currently reached position.

5.8.1.8 Command Online Change Memory Reserve Settings

Function: The command opens the **Online Change Memory Reserve** view.

Call: Menu **PLC > Window**.

This view is used to configure the memory reserves for the online change for function blocks.

Browse application	• Searches the selected PLC project for function blocks and displays them in the **Function blocks** area • Updates the **Function blocks** area after the PLC project has been compiled again • Updates the **Function blocks** area after an online change
Selection list with the PLC projects of the open TwinCAT project	Selection of the PLC project whose function blocks are to be displayed and/or edited in this view

Function blocks:

All	All function blocks of the selected PLC project are displayed.
No memory reserve	All function blocks with a memory reserve of 0 bytes are displayed.
<Memory reserve> bytes	Display of all function blocks with the number of bytes defined in **memory reserve**.
Information on the function blocks Multiple selection is also possible when selecting a function block for configuring the memory reserve.	
Function block	Name of the function block
Size	Size of the function block Size of an instance of the function block Specification in bytes
Number of instances	Number of instances of the function block in the project
Memory reserve	Displays the memory reserve for each instance of the function block
Additional memory for all instances	Product of **number of instances** and **memory reserve**
Remaining memory reserve	Number of bytes that are available as reserve per function block instance

Settings:

Memory reserve (in bytes)	Input field for the memory reserve for the selected function block. Specification in bytes Requirement: The PLC project is not yet on the controller or you have allowed the memory reserve to be changed by clicking the **Allow** button in the **Allow editing** area.
Apply to selection	The **memory reserve (in bytes)** is assigned to the function block, and the **Memory reserve** table column is updated. If multiple function blocks are selected, the entered value is assigned to each function block. To update the columns **Size**, **Number of instances**, **Additional memory for all instances**, and **Remaining memory reserve size**, first choose **Create > Create**, then click the **Browse application** button.

Enable editing:

Enable	The input field **Memory reserve (in bytes)** becomes editable. This button is changed to **Editable**.

Information:

Number of FBs	Total number of function blocks in the PLC project
Additional memory for all instances	Sum of the memory reserves of all function block instances of the PLC project Specification in bytes

See also:

- PLC documentation: Program PLC project > Configure memory reserve for online change

5.8.2 Core dump

5.8.2.1 Command Generate core dump

Function: The command causes TwinCAT to first check whether a core dump file is already available on the target system.

- If a core dump file is available on the target system, TwinCAT offers you to load this file into the project directory. There are three different ways to respond to the query as to whether the core dump file is to be loaded from the target system.
  - Yes: If the core dump file of the target system matches the currently logged-in PLC project, TwinCAT loads the core dump file from the target system to the project directory. You can open this file by subsequently logging out the PLC project and using the Command Load core dump [▶ 136].
  - No: A new core dump file will be created in the project directory. The prerequisite for this is that the PLC project is currently at a breakpoint or that an exception error has occurred.
  - Cancel: The creation of a core dump file is aborted.
- If no core dump file is available on the target system, TwinCAT initiates the creation of a new dump file with the current PLC project data in the project directory. The prerequisite for this is that the PLC project is currently at a breakpoint or that an exception error has occurred.

The core dump file that is created is stored directly in the PLC project directory: <PLC project name>.<PLC project GUID>.core

Call: Menu PLC > Core dump

Requirement: The PLC project is in online mode.

Automatic generation of a core dump on the target system

If the PLC project running on a target system is not currently logged into a development environment, the runtime system automatically generates a core dump on the target system in the event of an exception error. This file is located in the boot folder of the target system (by default under C:\TwinCAT\\3.1\Boot\Plc).

The automatic loading of this dump file from the target system to the local project directory is possible with the help of the **command Create Core Dump.** It is also possible to copy the core dump file from the target system to the development computer.

Displaying the dump using the command Load Core Dump [▶ 136] can thus be used for (subsequent) error analysis.

Core dump usable only with associated compile info file

If you archive or save a core dump file, please note that the associated project and associated compile info file (*.compileinfo file, which is stored, for example when creating the project, in the "_CompileInfo" folder) must be present in order to load a core dump. If this is not the case, TwinCAT cannot use the dump later.

Please also note here the setting options on the Settings tab [▶ 112]. With the help of the **Core Dump** setting, you can configure whether the core dump file, which may be located in the project directory, is to be saved together with the available compile info files in a TwinCAT file archive.

See also:

- PLC documentation: PLC project at runtime > Error analysis with core dump
- Command Load core dump [▶ 136]

5.8.2.2 Command Load core dump

Function: TwinCAT searches the project directory for core dump files.

- If TwinCAT finds a core dump file in the project directory, you are asked whether you want to load this core dump or browse for a dump file.
- If TwinCAT does not find a core dump file in the project directory, you can browse for another dump file.

Loading into the project causes an online view of the PLC project to appear, with the state the PLC project had at the time when the core dump was created. You can view the variable values contained in it. The call tree is also available.

Call: Menu PLC > Core dump

Requirement: The application is in offline mode.

The Core Dump view can only be closed again using the Command Close core dump [▶ 137] command. The Logout command is not effective in this view.

Core dump usable only with associated compile info file

If you archive or save a core dump file, please note that the associated project and associated compile info file (*.compileinfo file, which is stored, for example when creating the project, in the "_CompileInfo" folder) must be present in order to load a core dump. If this is not the case, TwinCAT cannot use the dump later.

Please also note here the setting options on the Settings tab [▶ 112]. With the help of the **Core Dump** setting, you can configure whether the core dump file, which may be located in the project directory, is to be saved together with the available compile info files in a TwinCAT file archive.

See also:

- PLC documentation: PLC project at runtime > Error analysis with core dump
- Command Generate core dump [▶ 135]
- Command Close core dump [▶ 137]

5.8.2.3 Command Close core dump

Function: The command closes the core dump view of the PLC project that is currently open in the development environment.

Call: Menu PLC > Core dump

Requirement: The PLC project is in offline mode and you have loaded a core dump file into the project.

See also:

- PLC documentation: PLC project at runtime > Error analysis with core dump

5.8.3 Command Download

Function: The command causes the active PLC project to be compiled and then downloaded to the controller.

Call: PLC menu

Requirement: The PLC project is in online mode.

With this command, TwinCAT performs a syntax check and generates the program code. This code is loaded onto the controller. TwinCAT also generates the compile log <projectname>.<devicename>.<application ID>.compileinfo in the project directory.

During the download, all variables except persistent variables are reinitialized.

The description of the **Login** command explains the possible situations when logging in and loading.

If you try to load a PLC project while the same version of this project is already on the controller, the following message appears: "The program is unchanged. Application was not loaded". TwinCAT does not load the project onto the PLC.

When loading, a log of the actions that have taken place (creating the code, carrying out initializations, etc.) is displayed in the **Output** view. In addition, information about memory areas, the size of the code, global data and allocated memory is displayed. For the sake of clarity, in contrast to online change, the changed function blocks are no longer listed.

See also:

- Command Login [▶ 139]

5.8.4 Command Online Change

Function: The command is used to initiate an online change to the currently active PLC project. TwinCAT only reloads the modified parts of a PLC project that is already running on the controller into the controller.

Call: **PLC** menu

Requirement: The PLC project is in online mode.

An online change is not possible after the **Clean All** and **Clean** commands. The cleanup process deletes the compile information (compile log) that is automatically saved each time code is generated, and which forms the basis for an online change.

⚠ WARNING

Damage to property and persons due to unexpected behavior of the machine or system

An online change modifies the running application program and does not cause a restart. Depending on the machine being controlled, the machine or workpieces may be damaged or the health and life of people may be endangered.

- Make sure that the new program code results in the desired behavior of the controlled system.

Project-specific initializations

When an online change is performed, the project-specific initializations (homing etc.) are not executed because the machine retains its status. For this reason, the new program code may not have the desired effect.

Major changes in the download code

If the online change causes significant changes in the download code (e.g. shifting of variables is required), a dialog provides information about the effects and allows you to cancel the online change.

Fast online change

For small changes (e.g. in the implementation section, with no shifting of variables required), a "fast online change" is performed. In this case, only the modified function block is compiled and reloaded. In particular, no initialization code is generated in this case. This also means that no code for initializing variables with the attribute 'init_on_onlchange' is generated. Usually this will not have any effect, since the attribute tends to be used to initialize variables with addresses, but a variable cannot change its address during a fast online change.

To ensure the init_on_onlchange attribute is applied to the entire application code, deactivate fast online change for the PLC project using the no_fast_online_change compiler definition. To this end, insert the definition in the Compile category [▶ 97] of the PLC project properties.

The attribute 'init_on_onlchange' has no effect for individual FB variables

The Attribute 'init_on_onlchange' only applies to global variables, program variables and local static variables of function blocks.

To reinitialize a function block during an online change, the function block instance must be declared with the attribute. The attribute is not evaluated for a single variable in a function block.

Pointer variables

Pointers retain their value from the last cycle. If a pointer points to a variable that was resized by the online change and moved in the memory as a result, the pointer variable no longer returns the correct position of the variable. Make sure that pointers are reassigned in each cycle.

During an online change with possibly unintended consequences, TwinCAT lists the changed interfaces, affected variables and all function blocks for which new code has been generated in the Details dialog box. If storage locations change, a dialog is displayed to indicate possible problems with pointers.

See also:

- PLC documentation: Programming a PLC project

5.8.5 Command Login

Symbol: 

Function: This command connects the programming system (the selected PLC project) with the target system (controller) and thus establishes online operation. An instance of the PLC project is created on the target system and loaded.

Call: **PLC** menu or **TwinCAT PLC toolbar options** or context menu of the PLC project object (<PLC project name>Project) in the **Solution Explorer**

Requirement: The PLC project is error-free and the target system is in Run mode.

Possible login situations:

- The PLC project does not yet exist on the controller: You will be prompted to confirm the download.
- The PLC project is already on the controller and has not been changed since the last download. Logging in takes place without further interaction with you.
- The PLC project is already on the controller, but has been changed since the last download. You will be prompted to choose one of the following options:
  - Login with Online Change (please refer to the notes in section "Command Online Change [▶ 138]")
  - Login with download
  - Login without changes

    At this point you can also update the boot project on the controller.
- An unknown version of the PLC project is already on the controller. You will be asked whether TwinCAT should replace it.
- A version of the PLC project is already running on the controller. You will be asked whether TwinCAT should nevertheless log in and overwrite the PLC program that is currently running.
- The PLC program on the controller is currently stopped at a breakpoint. You have logged out and changed the program: TwinCAT warns you that, in the event of an online change or download, the PLC will be completely stopped. This happens even if there are several tasks and only one of them is affected by the breakpoint.

Compiling the project before logging in

If a PLC project has not been compiled since its last change, TwinCAT compiles the project before logging in. This operation corresponds to the command **Compile in logged-out state**.

If errors occur during the compilation, a message dialog appears. The errors are displayed in the **Error List** view. You can then decide whether you want to log in without loading the program onto the controller.

See also:

- Command Build PLC project [▶ 115]

Error on login

If an error occurs while logging on to the controller, TwinCAT interrupts the loading process with an error message. The error dialog allows you to display the error details. If an exception error has occurred and the text *SOURCEPOSITION* is contained in the log message, you can use the command Display in Editor to display the relevant function in the editor. The cursor jumps to the line causing the error.

Output of information on the loading process

If TwinCAT loads the project onto the controller when logging in, the following information is displayed in the message window:

- Generated code size
- Size of global data
- Resulting memory requirement on the controller

- A list of the affected function blocks (for online change)

In online mode, you cannot change the settings of devices or modules. To change device parameters, you must log the PLC project out. Depending on the bus system, there may, however, be some special parameters that you can change in online mode.

TwinCAT stores the view configuration separately for online and offline mode. In addition, views that cannot be used in an operation mode are closed. For this reason, the view may change automatically when you log in.

5.8.6 Command Start

Symbol: 

Keyboard shortcut: **[F5]**

Function: The command starts the execution of the program.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode.

When you call the command from the **PLC** menu, it affects the currently active PLC project.

5.8.7 Command Stop

Symbol: 

Keyboard shortcut: **[Shift] + [F5]**

Function: The command stops the execution of the program.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode.

When you call the command from the **PLC** menu, it affects the currently active PLC project.

5.8.8 Command Logout

Symbol: 

Function: The command terminates the connection between the development system and the target system (controller or simulated device), thus switching to offline mode.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

5.8.9 Command Reset cold

Symbol: 

Function: The command resets all variables of the active PLC project to their initialization values, except the PERSISTENT variables and the RETAIN variables.

Call: Menu **PLC, TwinCAT PLC toolbar options**

Requirement: The PLC project is in online mode.

All variables of the active PLC project are reset, with the exception of the PERSISTENT variables and the RETAIN variables. The situation is the same as when an application program that has just been loaded onto the controller ("cold start") is started.

Breakpoints of the PLC program that were enabled before the active PLC project was reset are still enabled after execution of the command. Breakpoints that were previously disabled are still disabled after the command is executed.

If you select the command while the program is stopping at a breakpoint, you will be asked whether the current cycle is to be terminated. Alternatively, TwinCAT carries out the reset immediately. However, not all runtime systems are able to perform a reset without terminating the current cycle.

After the reset, you must start the PLC program with the **Start** command.

See also:

- PLC documentation: Remanent Variables - PERSISTENT, RETAIN
- PLC documentation: Resetting the PLC project
- Command Reset origin [▶ 141]
- Command Start [▶ 140]

5.8.10 Command Reset origin

Symbol: 

Function: The command resets all variables of the active PLC project, including the remanent variables (RETAIN, PERSISTENT variables), to their initialization values and deletes the application program from the controller.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode.

A reset deactivates the currently set breakpoints in the program. If you select the command while the program is stopping at a breakpoint, you will be asked whether the current cycle is to be terminated. Alternatively, TwinCAT carries out the reset immediately. However, not all runtime systems are able to perform a reset without terminating the current cycle.

See also:

- Command Reset cold [▶ 140]

5.8.11 Command Single cycle

Symbol: 

Function: The command executes the active PLC program for one cycle.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode and the program is at a program step.

5.8.12 Command Flow Control

Symbol: 

Function: The command activates or deactivates the flow control.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode.

An active flow control extends the runtime of the PLC project!

See also:

- PLC documentation: Flow Control

5.8.13 Command Force values

Symbol: 

Function: The command permanently sets the value of a variable on the controller to a predefined value.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode.

⚠ CAUTION

Material damage and personal injury due to unexpected behavior of the controlled system

Changing the values of variables in a PLC program running on the controller can lead to unexpected behavior of the controlled system. Depending on the controlled system, damage to equipment and workpieces can occur or people's health and lives may be endangered.

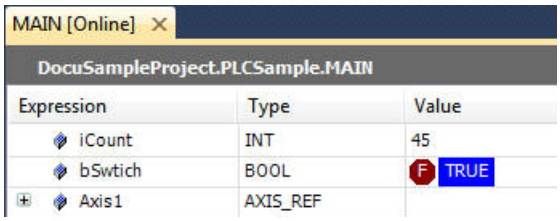
- Evaluate possible risks before forcing variable values and take appropriate safety precautions.

With this command, TwinCAT permanently sets one or more variables of the active application on the controller to defined values. This setting is carried out at the beginning and end of each processing cycle. Order of execution: 1. Read inputs, 2. Force values, 3. Execute code, 4. Force values, 5. Write outputs.

You can prepare values by

- clicking in the **prepared value** field in the declaration part and entering the new value. For a Boolean variable, change the value by single-clicking in the field.
- Click the inline monitoring field in the implementation part of the FBD/LD/IL editor and enter the new value.
- Click in the **prepared value** field in the Monitor window and enter the new value.

A "forced" value is marked with an F.



TwinCAT performs forcing until it is explicitly cancelled by the user through

- the command **Unforce values**
- cancellation of forcing via the **Prepare Value** dialog
- Logging out of the application

The command **Force values for all applications**, which affects all PLC projects in the TwinCAT project, is not included in a menu by default.

See also:

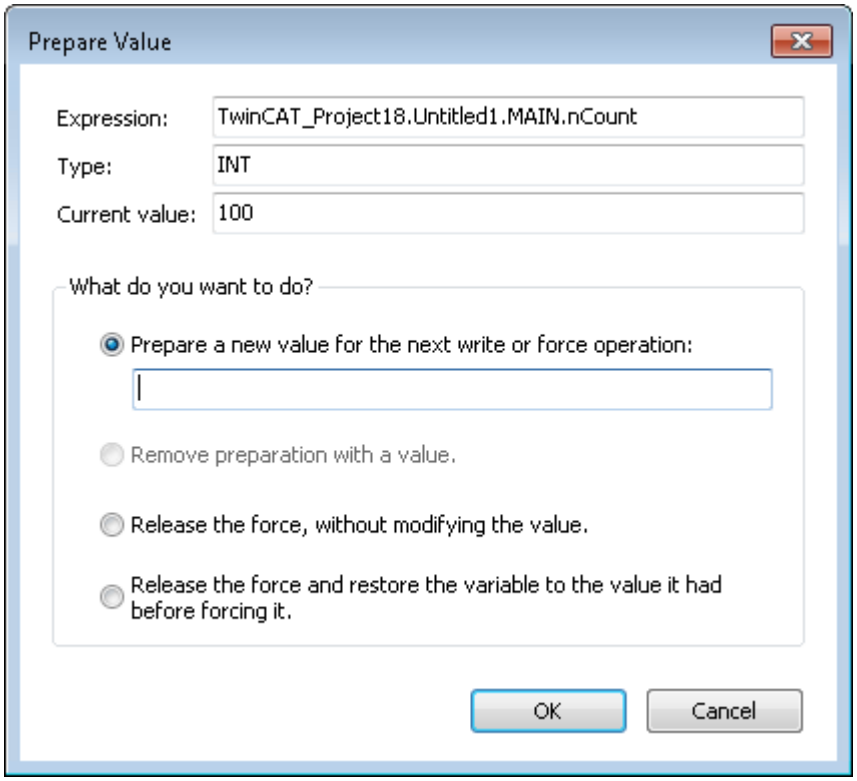
- Command Unforce values [▶ 143]
- Dialog Prepare Value [▶ 143]
- PLC documentation: Forcing and Writing Variables Values

5.8.13.1 Dialog Prepare Value

Function: The dialog is used to prepare a value for an already forced variable. TwinCAT executes the prepared action with the next forcing.

Call: TwinCAT opens the dialog in the following situations:

- if you click in the field **prepared value** of a forced variable in the declaration part
- if you click in the inline monitoring field of a forced variable in the implementation part
- if you click in the **prepared value** field of a forced variable in the Monitor window



Preparing a new value for the next write or force operation	Value that TwinCAT writes to the variable during the next force operation
Remove preparation with a value	TwinCAT deletes the prepared value.
Release the force, without modifying the value.	TwinCAT retains the forced value and terminates forcing. TwinCAT marks the variable with <Unforce>.
Release the force and restore the variable to the value it had before forcing it.	TwinCAT resets the forced value and terminates forcing. The variable is marked with <Unforce and restore>.

See also:

- Command Force values [▶ 142]

5.8.14 Command Unforce values

Symbol:

Function: The command resets the forcing of all variables. The variables get their current value from the controller.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode.

⚠ CAUTION

Material damage and personal injury due to unexpected behavior of the controlled system

Changing the values of variables in a PLC program running on the controller can lead to unexpected behavior of the controlled system. Depending on the controlled system, damage to equipment and workpieces can occur or people's health and lives may be endangered.

- Evaluate possible risks before resetting forced variable values and take appropriate safety precautions.

The command **Unforce values for all applications**, which affects all PLC projects in the TwinCAT project, is not included in a menu by default.

See also:

- Command Force values [▶ 142]
- PLC documentation: Forcing and Writing Variables Values

5.8.15 Command Write values

Symbol:

Function: The command once sets the value of a variable on the controller to a predefined value.

Call: Menu **PLC, TwinCAT PLC Toolbar Options**

Requirement: The PLC project is in online mode.

⚠ CAUTION

Material damage and personal injury due to unexpected behavior of the controlled system

Changing the values of variables in a PLC program running on the controller can lead to unexpected behavior of the controlled system. Depending on the controlled system, damage to equipment and workpieces can occur or people's health and lives may be endangered.

- Evaluate possible risks before writing variable values and take appropriate safety precautions.

With this command, you set one or more variables of the active PLC project on the controller to defined values once. Writing takes place once at the beginning of the next cycle.

You can prepare values by

- clicking in the **prepared value** field in the declaration part and entering the new value. For a Boolean variable, change the value by single-clicking in the field.
- Click the inline monitoring field in the implementation part of the FBD/LD/IL editor and enter the new value.
- Click in the **prepared value** field in the Monitor window and enter the new value.

The command **Write values for all applications**, which affects all PLC projects in the TwinCAT project, is not included in a menu by default.

See also:

- Command Force values [▶ 142]
- PLC documentation: Forcing and Writing Variables Values

5.8.16 Command Display Mode - Binary, Decimal, Hexadecimal

Function: The commands of the **Display** submenu are used to set the format for displaying the values during monitoring in online mode.

Call: PLC menu

Requirement: The PLC project is in offline or online mode.

The display formats "Binary" and "Hexadecimal" are unsigned, "Decimal" is signed.

See also:

- PLC documentation: Monitoring Values

5.8.17 Command Create Localization Template

Function: The command opens the **Create Localization Template** dialog. Here you define which text information from the project is to be exported to a translation template of the file format pot.

Call: Menu **PLC > Project Localization**

Requirement: A project is open.

Create Localization Template dialog

The dialog is used to select the textual information to be included in the localization template.

Include the following information

Names	Texts such as dialog title, object names in the PLC project tree
Identifier	Variable identifier, example: "nCounter"
Strings	For example, "count" in the following declaration: sVar : STRING := 'count'
Comments	Comment texts in the programming blocks

Position information	Select which positions of the text categories in the project selected above should be included in the translation file. The position information is always in the first line(s) of a section for a translation.
	Example:
	`#: D:\Proj1.project\Project_Settings:1`
	`msgid „Projekteinstellungen“`
	`msgstr „„`
	• "All": All found positions of the text are listed.
	• "First occurrence": The translation file includes the position in the project where the text to be translated occurs for the first time.
	• "None"

Create	The button opens the dialog for saving a file. The translation template is created in a text file of type POT Translation Template (*.pot). Each further create process generates a complete new template file.

5.8.18 Command Manage Localization

Function: The command opens the **Manage Localization** dialog. In the dialog, select the desired localization language or the original version of the project. You can also add or remove localization files *.<Language>.po to or from the project.

Call: Menu **PLC > Project Localization**

Requirement: A project is open.

Manage localization dialog

Available localizations	List of the localization files present in the project.
	Example:
	`proj1-de.po`
	`proj1-en.po`
	`<Originalversion>`
	The original version is always available. The project can only be edited in the original version.

Add	The button opens the dialog for selecting another po file from the file system.
Remove	The button removes the po file selected on the left from the project.
Standard localization	 The currently selected localization becomes the standard localization. The entry is displayed in bold.
Change localization	Use the button to switch to the currently selected localization.
OK	The project is displayed in the language of the country supplied by the file selected in Files. If you select <**Original version**>, the project will appear in the editable, unlocalized version.

5.8.19 Command Toggle Localization

Symbol: 

Function: The command switches between the currently set project localization and the <Original version>.

Call:

- Menu **Project > Localization**
- Button in the **Manage Localizations** dialog
- Button in the toolbar

Requirement: A project is open. A standard localization for the project is defined in the **Manage Localizations** dialog.

See also:

- Command Manage Localization [▶ 145]

5.8.20 Command Active PLC project

Function: Drop-down list for selecting the active PLC project.

Call: TwinCAT PLC Toolbar Options

5.8.21 Command Active PLC instance

Function: Drop-down list for selecting the active PLC instance of the corresponding PLC project.

Call: TwinCAT PLC Toolbar Options

5.9 Tools

5.9.1 Command Options

Function: The command opens the **Options** dialog for configuring TwinCAT options. These options define the behavior and appearance of the TwinCAT user interface. TwinCAT saves the current settings on the local system as default settings.

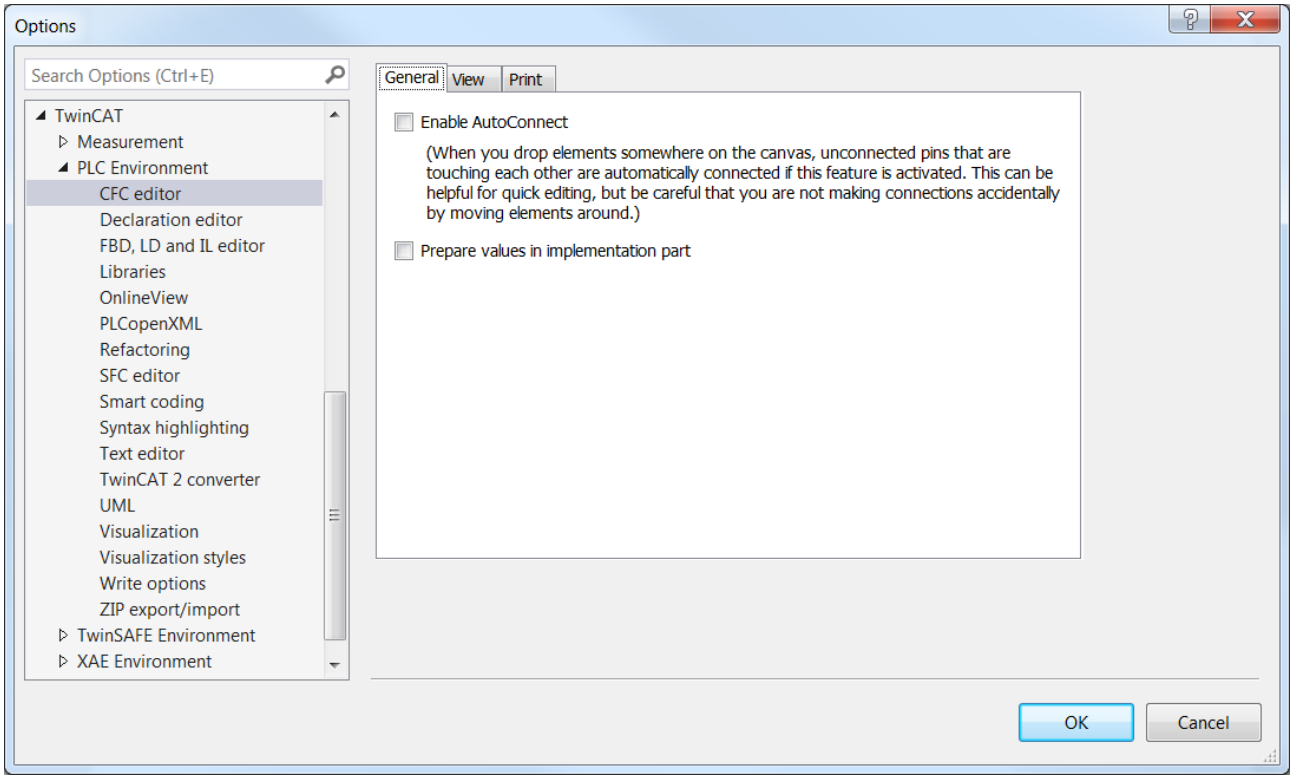
Call: Menu **Tools**

5.9.1.1 Dialog Options - CFC Editor

Function: The dialog is used to configure the settings for editing and printing in the CFC editor.

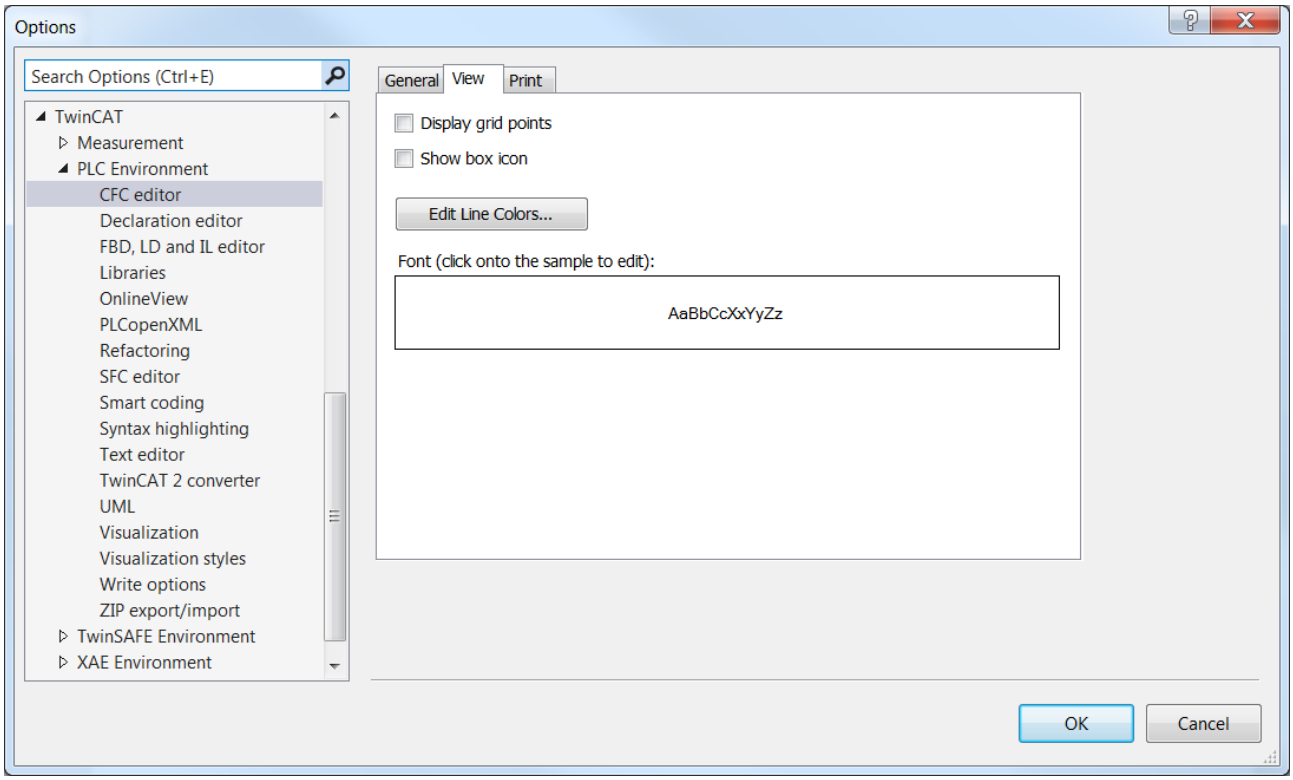
Call: TwinCAT > PLC Environment > CFC editor

General tab



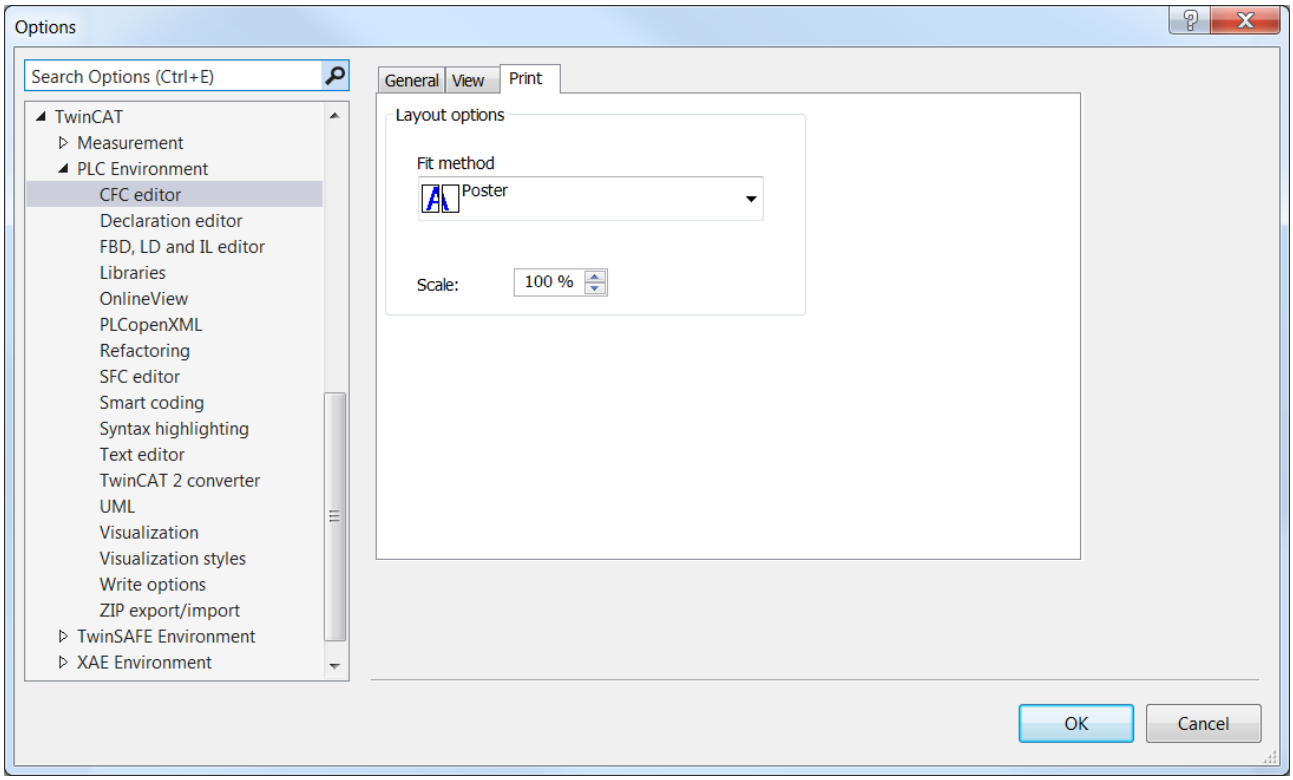
Enable AutoConnect	☑ : When you drag and paste a CFC element onto the editor's workspace, TwinCAT automatically connects unlinked pins that "touch" each other. Make sure that you do not create any unwanted connections when you move elements!
Prepare values in implementation part	☑ : In online mode, you can also prepare variable values for writing and forcing in the implementation part of the CFC function block. In addition, TwinCAT displays the values that have just been prepared in the inline monitoring box of the variables in angle brackets.

View tab



Display grid points	☑ : In the editor, grid points are validity areas at which you can position the elements.
Show box icon	☑ : TwinCAT displays function blocks in the CFC editor that are linked to a bitmap as symbols. Requirement: You have either created the link for a function block or a function in the object properties or loaded it using a library.
Edit Line Colors	Opens the **Edit Line Colors** dialog to define the colors for the connection lines depending on the current data type. The lines appear in these colors in offline and online mode, except TwinCAT overpaints these colors with bold black and blue lines to indicate Boolean data flow. • Add type: Adds a data type to the list. • Remove type
Font	Displays the font and button for changing the font.

Print tab



Layout options

Fit method	Page or poster
Scale	Possible values: 20% - 200%

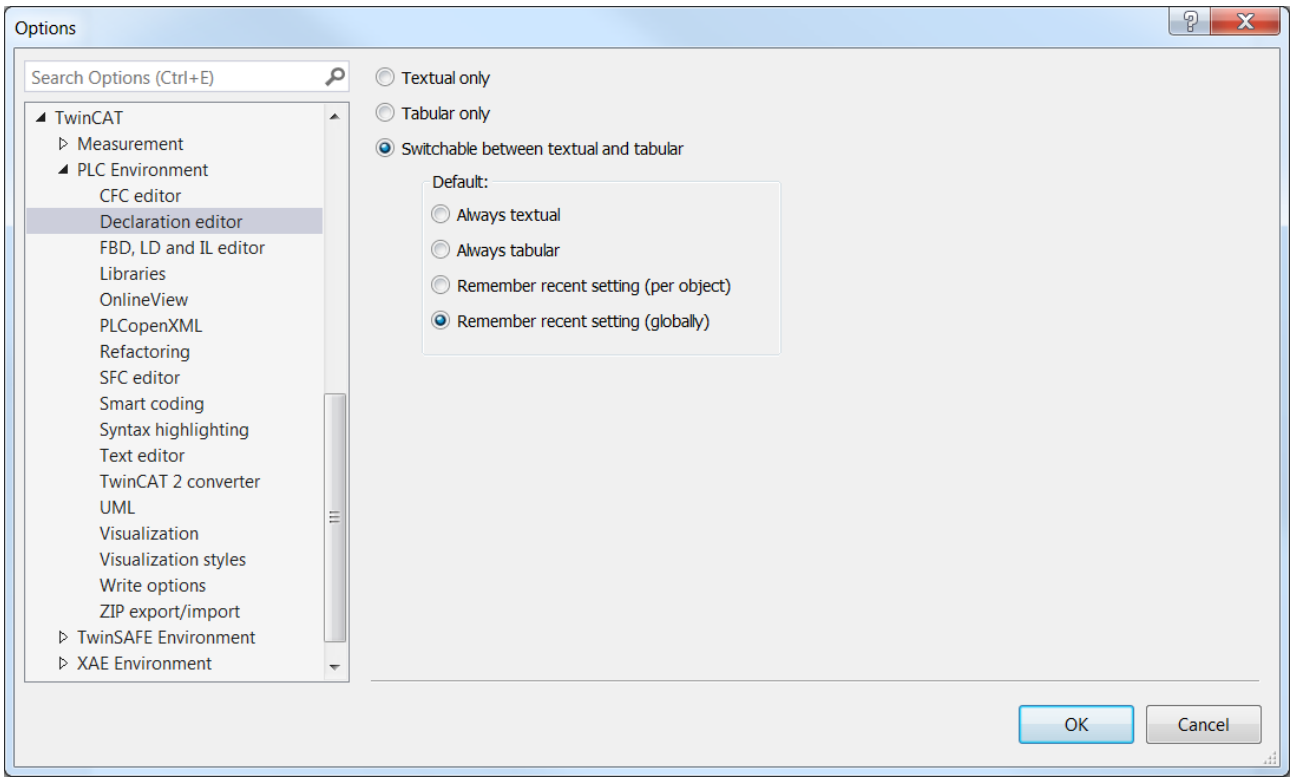
See also:

- PLC documentation: Programming in CFC
- PLC documentation: Programming languages and their editors

5.9.1.2 Dialog Options - Declaration Editor

Function: The dialog is used to configure the display settings for the declaration editor.

Call: TwinCAT > PLC Environment > Declaration editor



Textual only	Textual view of the declaration editor
Tabular only	Tabular view of the declaration editor
Switchable between textual and tabular	The declaration editor provides two buttons to switch between the textual and tabular views: : Textual view : Tabular view The following option defines the view that appears by default when you open a programming object: • Always textual • Always tabular • Remember recent setting (per object) • Remember recent setting (globally)

See also:

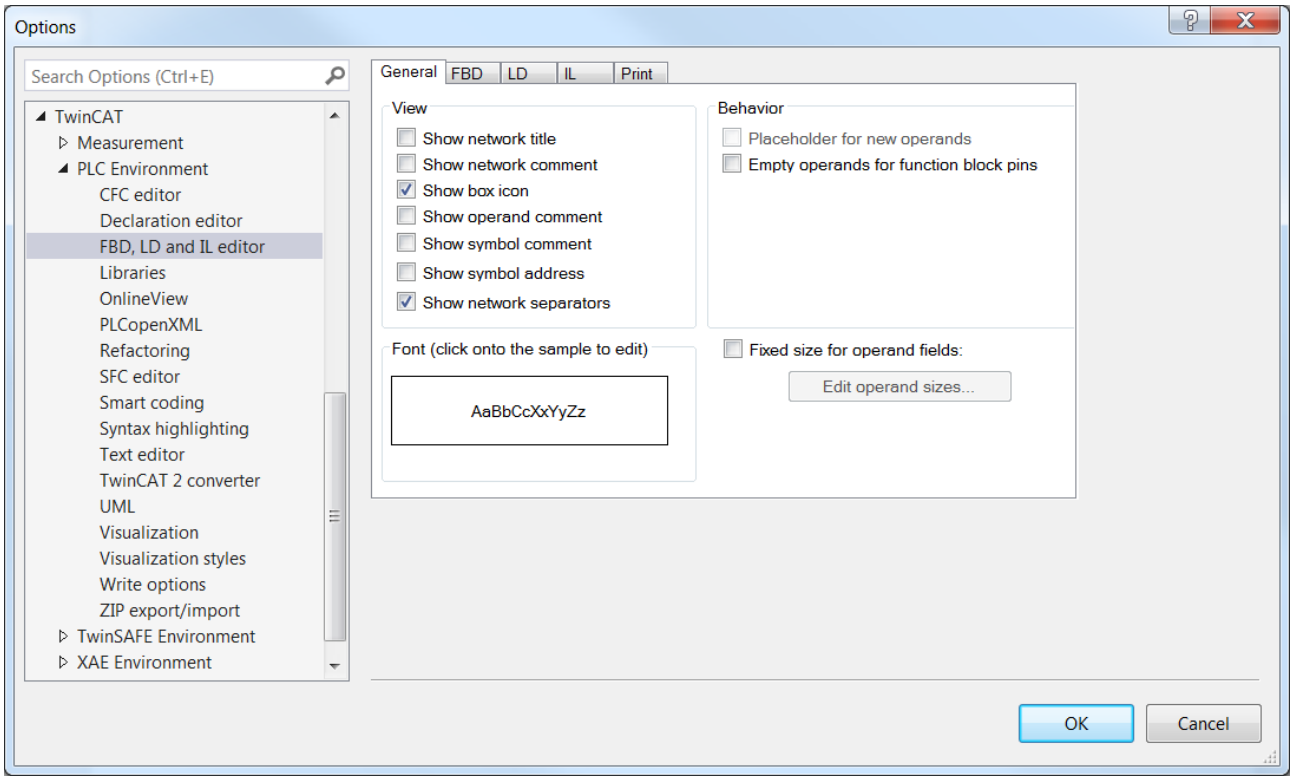
- PLC documentation: Using the declaration editor

5.9.1.3 Dialog options - FBD, LD and IL

Function: The dialog is used to configure the presentation options for the FBD/LD/IL editor.

Call: TwinCAT > PLC Environment > FBD, LD and IL

General tab



View

Show network title	The network title is displayed in the upper left corner of the network.
Show network comment	The network comment is displayed in the upper left corner of the network. If TwinCAT also displays the network title, the comment appears in the line below.
Show box icon	The function block symbol is displayed in the block element in the FBD and LD editor. The standard operators also have symbols.
Show operand comment	TwinCAT displays the comment that you have assigned to a variable in the implementation part. The operand comment only refers to the local usage point of the variable, in contrast to the "symbol comment".
Show symbol comment	TwinCAT displays the comment that you have assigned to a variable or symbol in the declaration above the variable name. In addition to or instead of the symbol comment, you can also assign a local "operand comment".
Show symbol address	If an address is assigned to a symbol (variable), this address is displayed above the variable name.
Show network separators	A dividing line is displayed between the individual networks.

Behavior

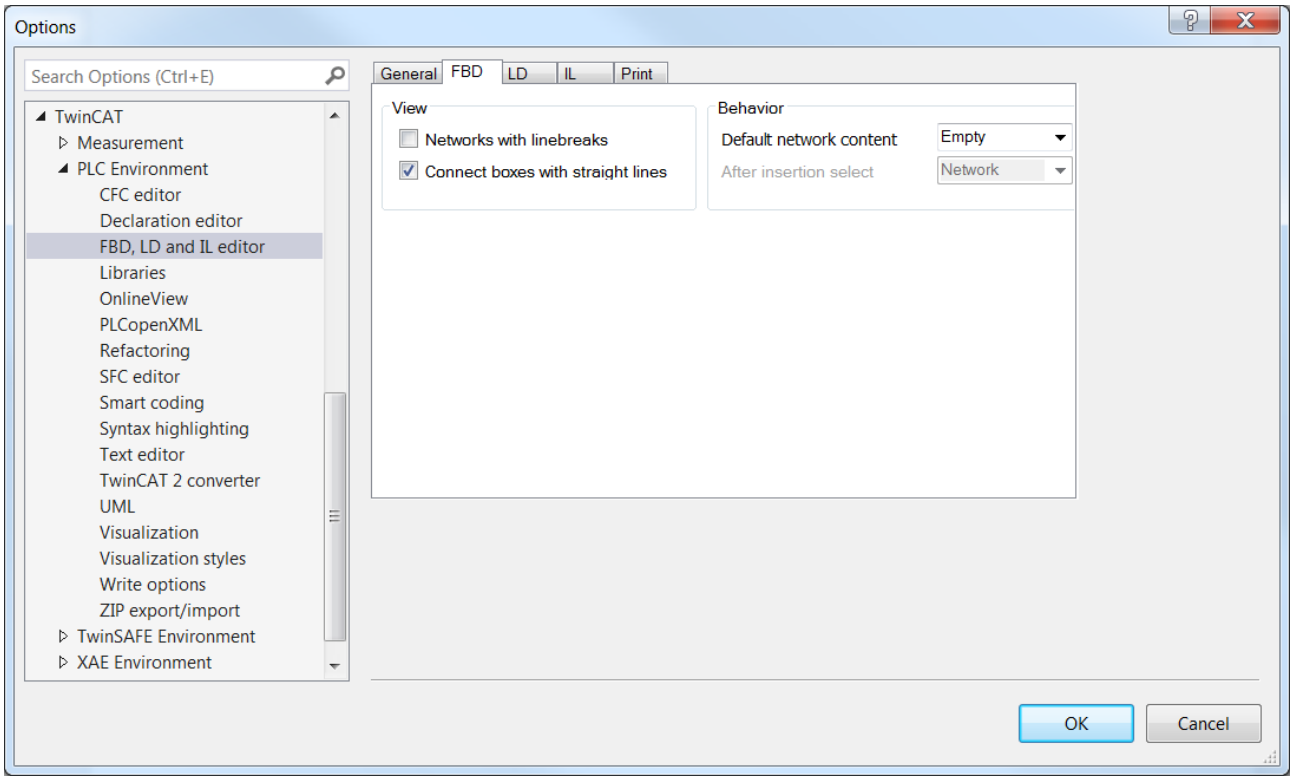
Placeholder for new operands	The operand field of the pins of the new function blocks is left blank (instead of "???").
Empty operands for function block pins	Inserts empty operands instead of ???.

Font

Clicking on the input field opens the **Font** dialog.

Fixed size for operand fields	☑ : Edit operand sizes can be enabled.
Edit operand sizes	The **Operand Size** dialog appears for setting the number of characters and lines.

FBD tab



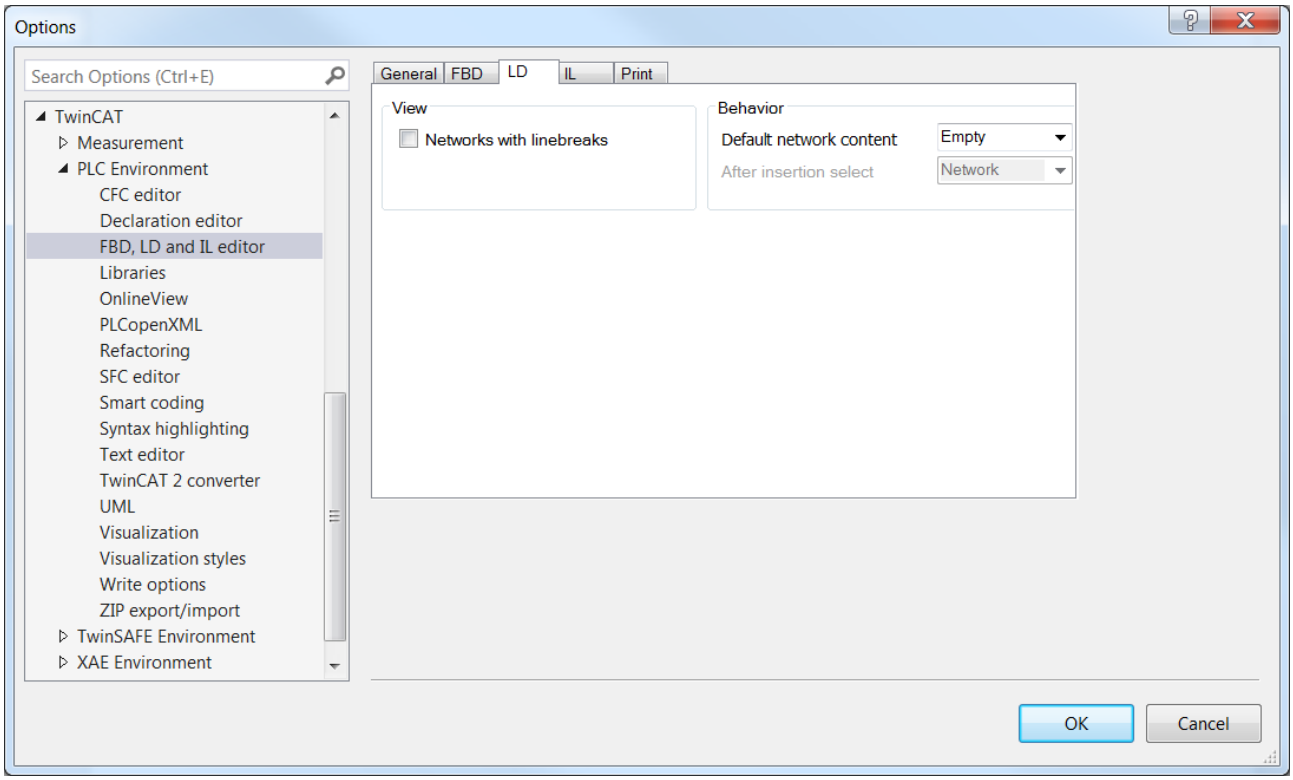
View

Networks with linebreaks	☑ : Representation of the networks with line breaks, so that TwinCAT can display as many function blocks as possible within the current width of the window.
Connect boxes with straight lines	☑ : Lines between the elements are given a fixed, short length.

Behavior

Default network content	Selection list: Contents of a new network.
After insertion select	Selection list: Element that TwinCAT selects after inserting a new network.

LD tab



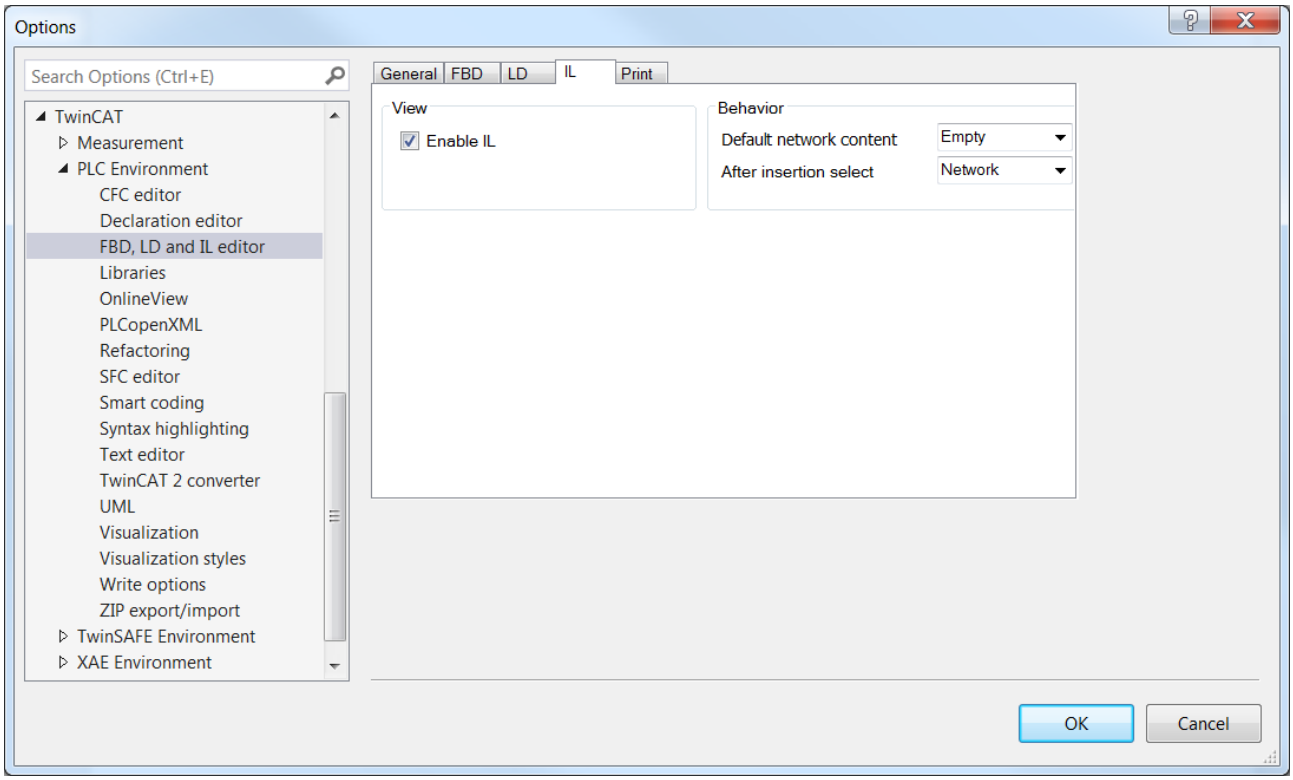
View

Networks with linebreaks	☑ : Representation of the networks with line breaks, so that TwinCAT can display as many function blocks as possible within the current width of the window.

Behavior

Default network content	Selection list: Contents of a new network.
After insertion select	Selection list: Element that TwinCAT selects after inserting a new network.

IL tab



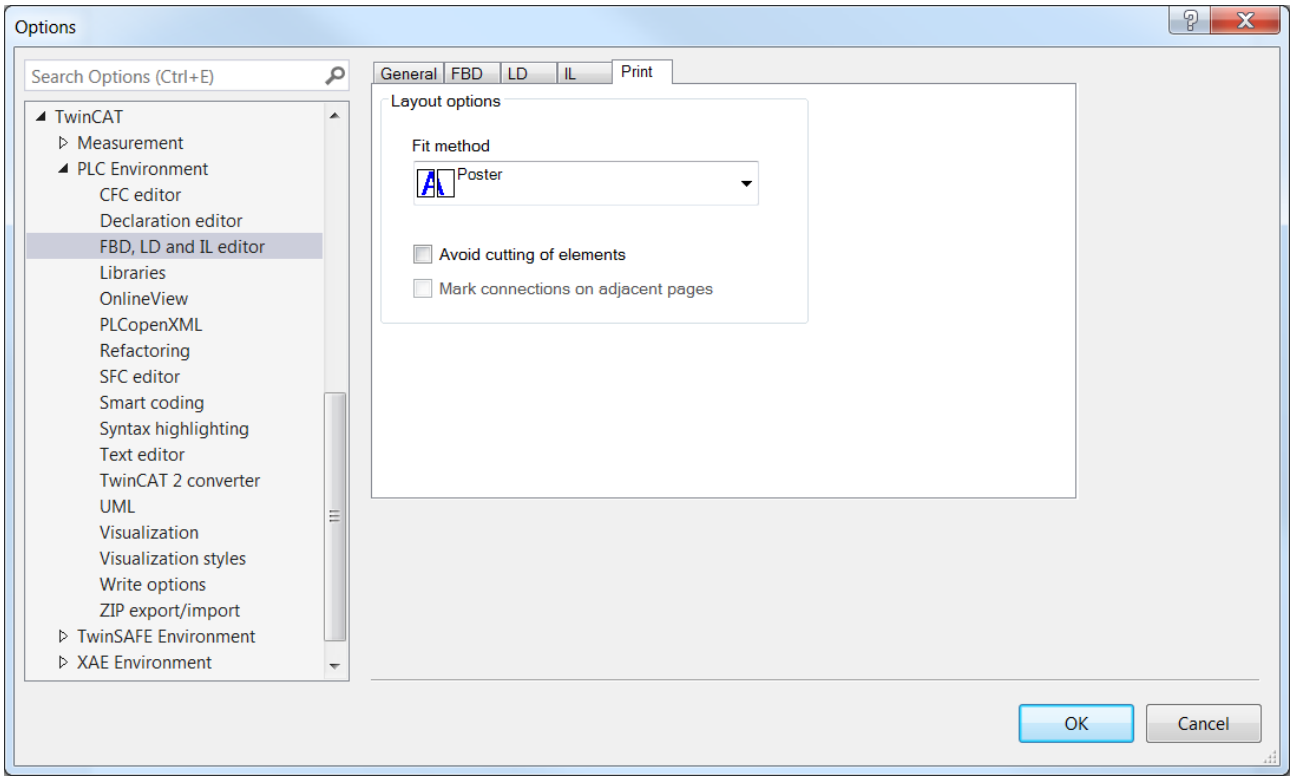
View

Enable IL	The implementation language IL is available in the development system.

Behavior

Default network content	Selection list: Contents of a new network.
After insertion select	Selection list: Element that TwinCAT selects after inserting a new network.

Print tab



Layout options

Fit method	Selection list for size adjustment.
Avoid cutting of elements	Items that do not fit on the page are printed on the next page.
Mark connections on adjacent pages	Can be enabled if **Avoid cutting of elements** is enabled.

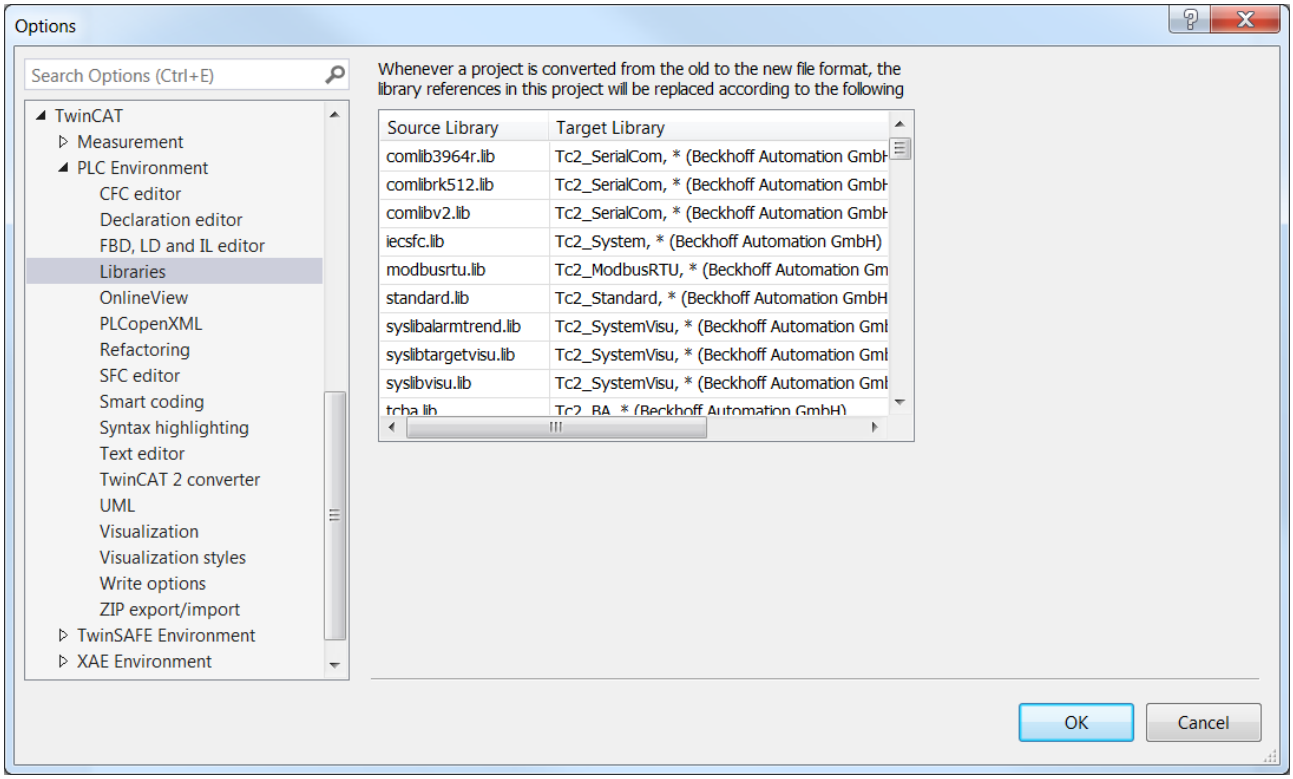
See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: Programming languages and their editors

5.9.1.4 Dialog Options - Libraries

Function: The dialog is used to manage the mappings of library references that TwinCAT uses when converting an old project. If you have not yet saved a mapping for a particular library, you must redefine the mapping each time you open an old project that includes this library.

Call: TwinCAT > PLC Environment > Libraries



A mapping defines how a library reference looks like after conversion of the project to the current format. There are three options:

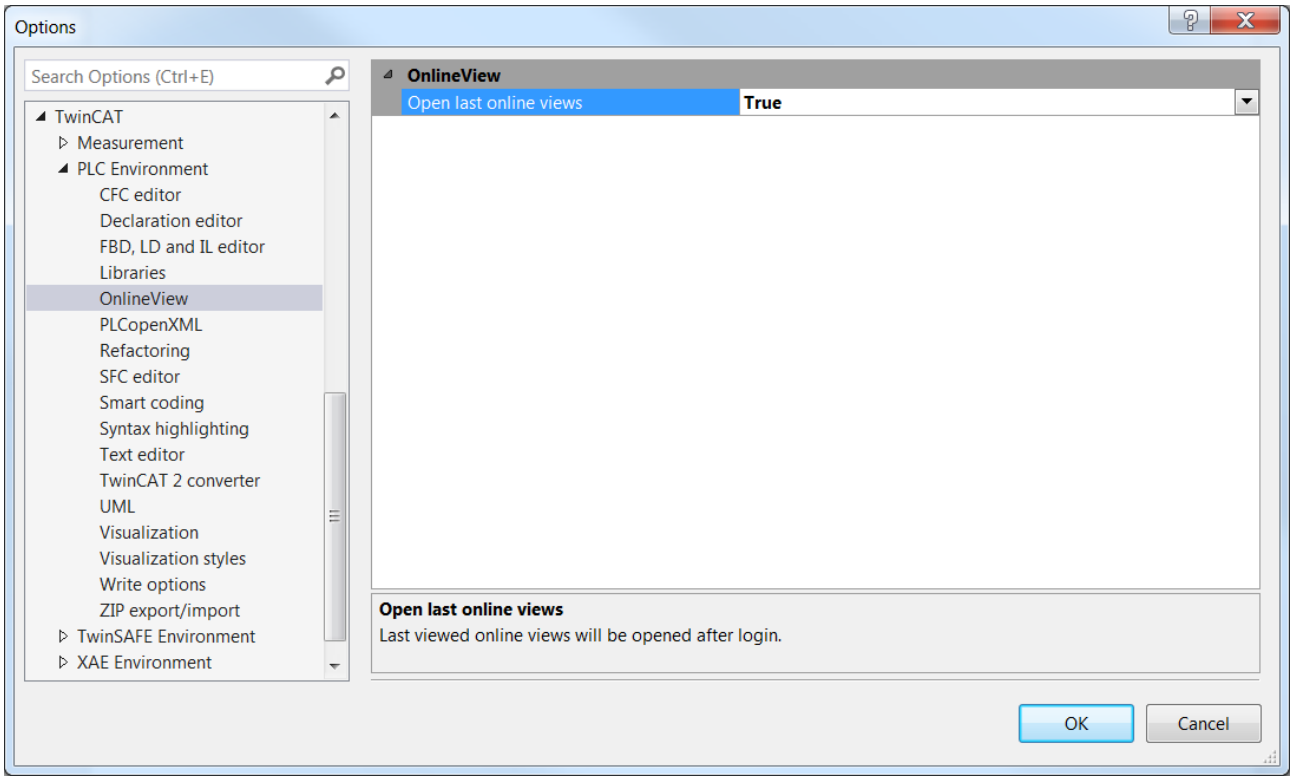
- You retain the reference. This means that TwinCAT also converts the library into the current format (*.library) and installs it in the local library repository.
- You replace a reference with another reference. This means that one of the installed libraries replaces the previously referenced library.
- You delete the reference. This means that the converted project no longer integrates the library.

During the next conversion of an old project, TwinCAT applies all listed mappings to its library references. This means that you have to repeat the mapping definition if the same library is included again in a project to be converted. You can enter a new mapping in the last line.	
Source Library	Path of the library that is included in the project before the conversion. Double-clicking an entry makes the field editable and the Input Assistant button appears.
Target Library	Name and location of the library to be included in the project after conversion. Double-clicking on an entry opens the "Set target system library" dialog.

See also:

- PLC documentation: Using libraries

5.9.1.5 Dialog Options - Online View



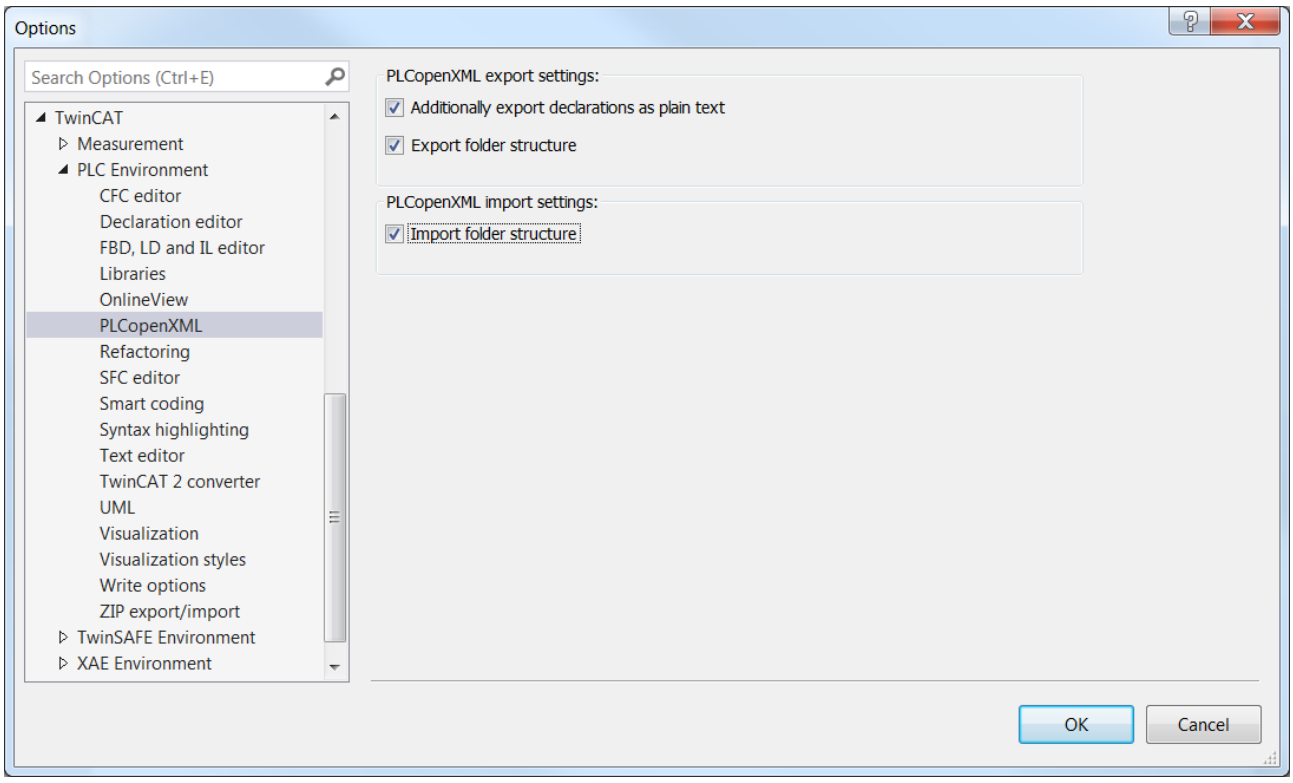
OnlineView

Open last online views	• TRUE (default setting): On login the editor windows of the previous online session open. The current offline view remains open. • FALSE: On login the offline view remains open. The editor windows of the previous online session are discarded and not reopened.

5.9.1.6 Dialog Options - PLCopenXML

Function: The dialog contains settings for the behavior of TwinCAT during PLCopenXML export or import.

Call: TwinCAT > PLC Environment > PLCopenXML



PLCopenXML export settings

Additionally export declarations as plain text	By default, TwinCAT splits the declaration parts into individual variables in accordance with the PLCopenXML schema, resulting in loss of formatting and some commentary information. ☑ : Formatting and comments are retained. TwinCAT also writes the plain text of the exported declaration part to the PLCopenXML file, thus extending the PLCopenXML schema.
Export folder structure	☑ : TwinCAT also exports the folders if they contain one of the selected objects. This is a TwinCAT-specific extension to the PLCopenXML schema.

PLCopenXML import settings

Import folder structure	☑ : If the import file contains information about the folder structure of the objects, TwinCAT imports this structure. ☐ : TwinCAT imports objects without a structure.

See also:

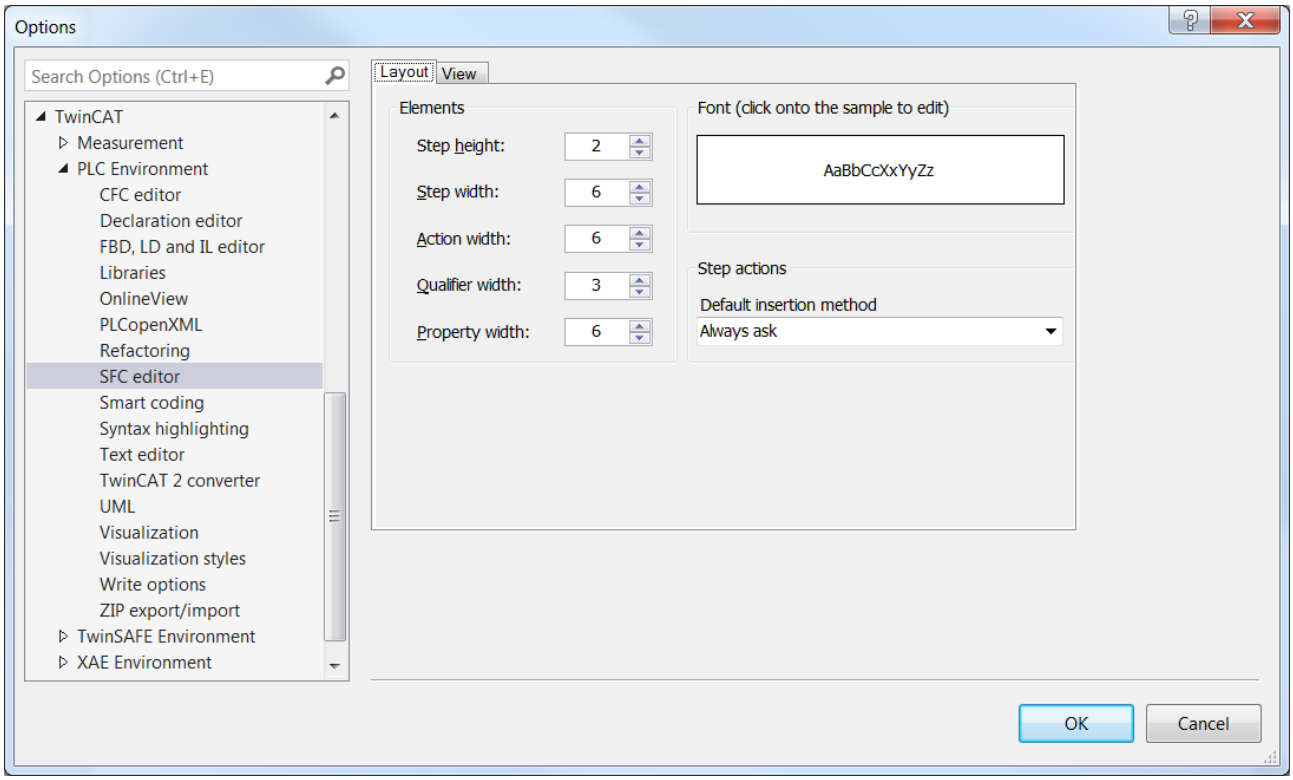
- Command Export PLCopenXML [▶ 245]
- Command Import PLCopenXML [▶ 245]
- PLC documentation: Exporting and importing a PLC project

5.9.1.7 Dialog Options - SFC editor

Function: The dialog is used to configure the settings for the SFC editor.

Call: TwinCAT > PLC Environment > SFC editor

Layout tab



Elements

Defines the sizes of the SFC elements Step, Action, Qualifier and Property. Specify the values in grid units. 1 grid unit = font size currently set in the text editor options (Text area / Font). The settings always take effect immediately in all currently open SFC editor windows.	
Step height	Possible values: 1-100
Step width	Possible values: 2-100
Action width	Possible values: 2-100
Qualifier width	Possible values: 2-100
Property width	Possible values: 2-100

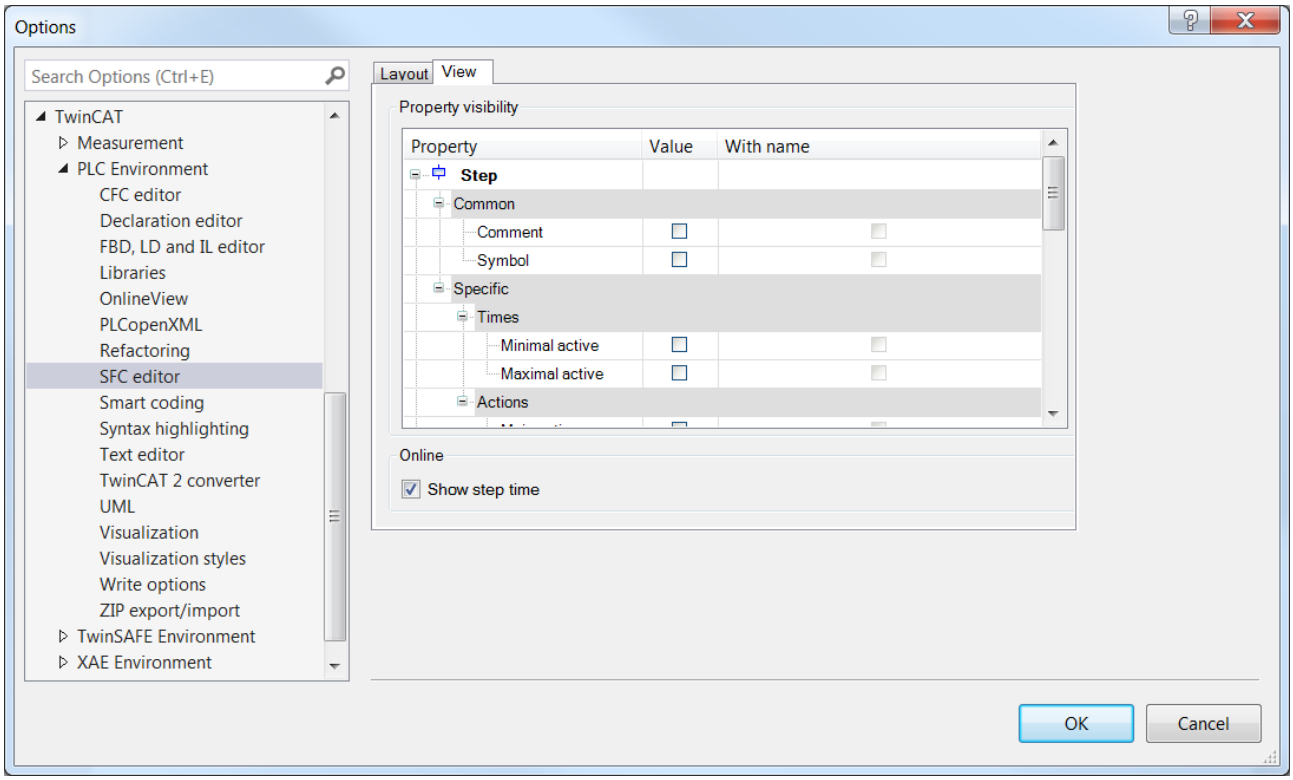
Font

The sample text shows the currently set font. Click on it to change the font.

Step actions

Default insertion method	• Always ask • Duplicate implementation • Copy reference

View tab



Property visibility

Lists the item properties of the Common and Specific categories and defines display options.	
Property	Defines the item properties that are shown next to the item in the SFC diagram.
Value	☑ : Display of the property value.
With name	☑ : Display of the property value with name.

Online

Show step time	☑ : TwinCAT displays the step time in online mode to the right of the steps.

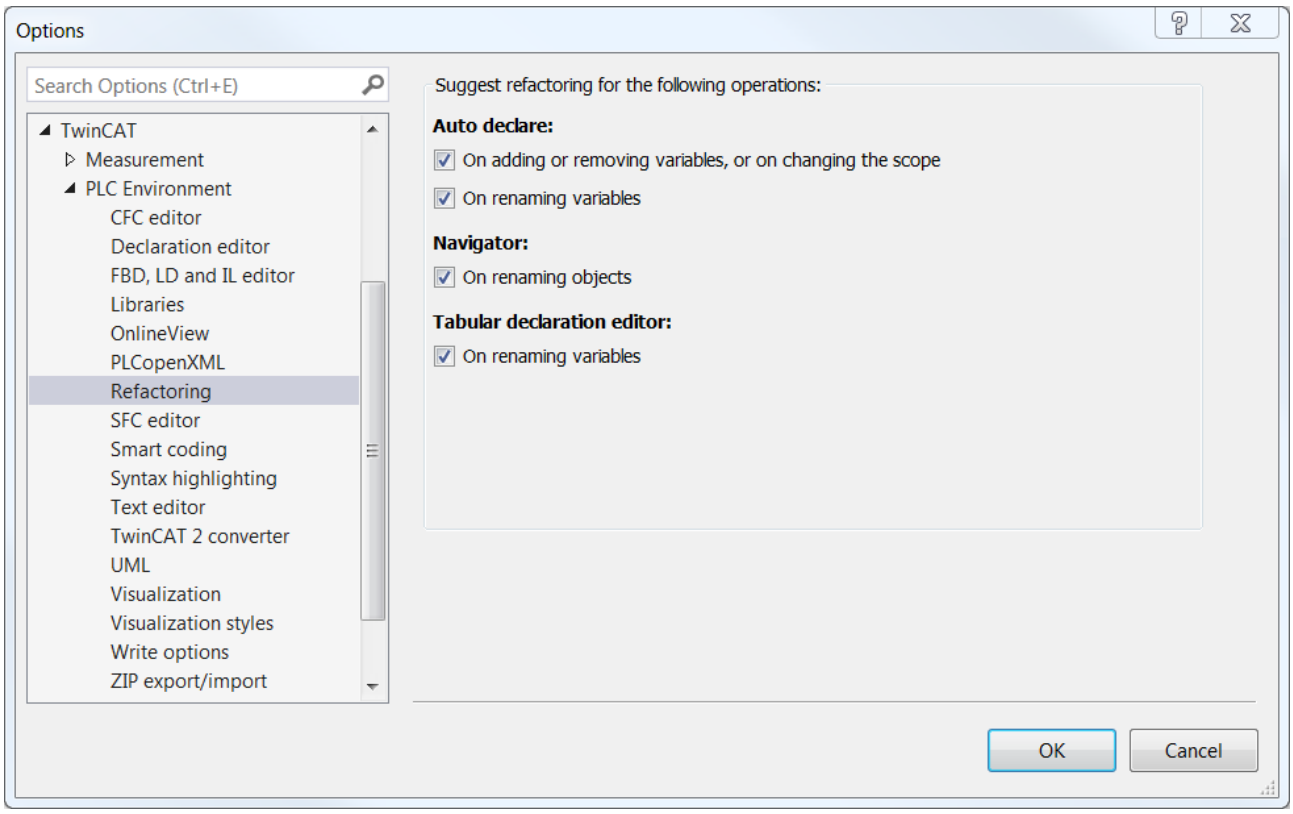
See also:

- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: Programming languages and their editors

5.9.1.8 Dialog Options - Refactoring

Function: The dialog is used to define the operations in the project for which refactoring is automatically proposed. The refactoring functionality supports your enhancement efforts.

Call: **TwinCAT > PLC programming environment > Refactoring**



Auto declare	If you change the name of a variable or add an input or output variable by calling the autodeclaration (**Auto Declare** dialog), the option **Apply changes using refactoring**, which appears during editing, is automatically enabled. After confirming the dialog, the **Refactoring** dialog opens and you can change the variable project-wide. ☑ (default setting) **On renaming variables** Rename the name in the auto declaration (**Auto Declare** dialog) and close the dialog with **OK**. This opens the **Refactoring** dialog to rename the variable project-wide. ☑ (default setting) **On adding or removing variables, or on changing the scope** Add a new input or output variable via the autodeclaration or delete the name of a variable in the autodeclaration (**Auto Declare** dialog) and close the dialog with **OK**. This will open the **Refactoring** dialog to remove the variable project-wide.
Tabular declaration editor	☑ (default setting) **On renaming variables** If you change the name of a variable in the tabular declaration editor, a prompt appears asking whether TwinCAT should carry out "Automatic refactoring" for the renaming.
Navigator	☑ (default setting) **On renaming objects** If you change the name of an object in the PLC project tree, a prompt appears asking whether TwinCAT should carry out “Automatic refactoring”.

See also:

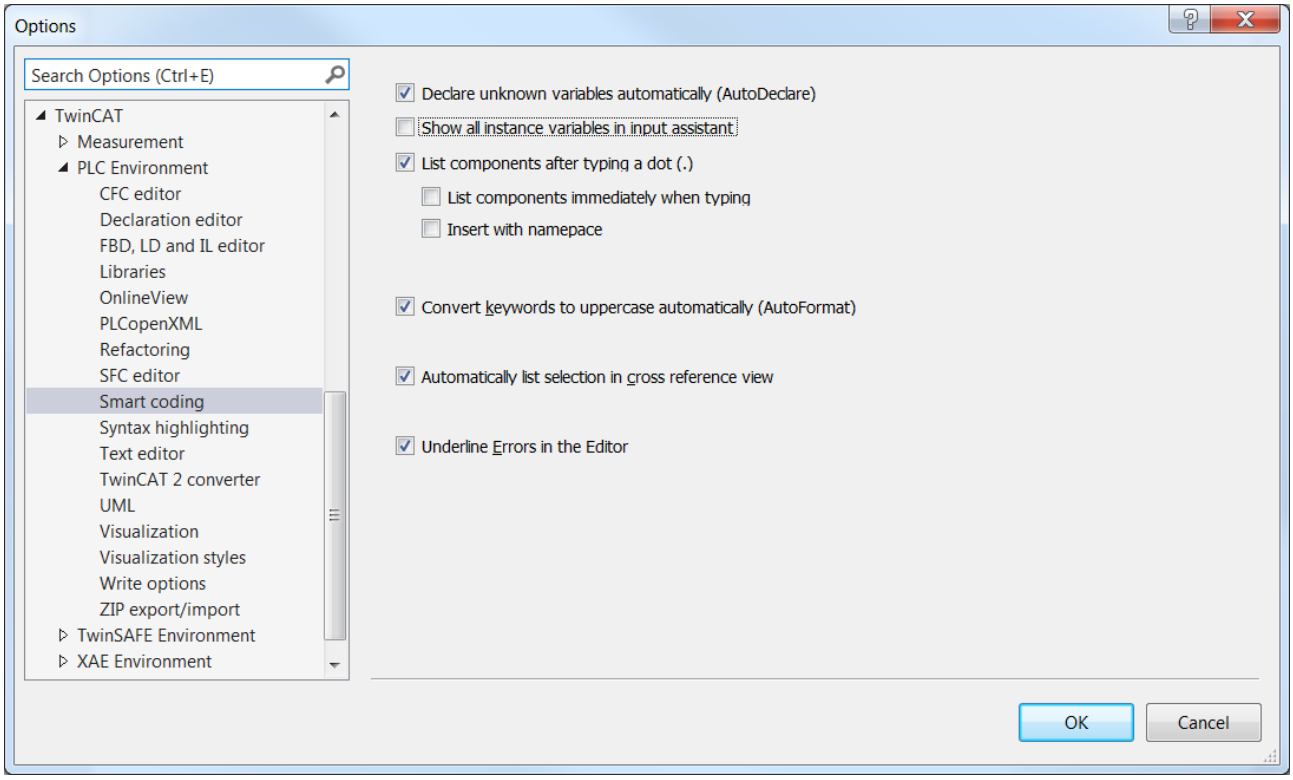
- Command Auto Declare [▶ 65]
- Command Rename '<variable>' [▶ 74]
- Command Add '<variable>' [▶ 75]
- Command Remove '<variable>' [▶ 77]

- PLC documentation: Refactoring

5.9.1.9 Dialog Options - Smart coding

Function: The dialog is used to configure the settings that make it easier to enter code.

Call: TwinCAT > PLC Environment > Smart Coding



Declare unknown variables automatically (AutoDeclare)	☑ : The **Auto Declare** dialog opens as soon as you have entered an undeclared identifier in a programming language editor and exited the input line.
Show all instance variables in input assistant	☑ : The **List components** function also offers the local variables of a function block instance for selection. ☐ : The **List components** function only offers the input and output variables of an FB instance for selection.
List components after typing a dot (.)	☑ : Enables the **List components** function. This means: If you enter a dot at a point where TwinCAT expects an identifier, a selection list with input options appears.
List components immediately when typing	Requirement: Option List components after typing a dot (.) is enabled. ☑ : After entering any character string, a selection list of the available identifiers and operators appears
Insert with namespace	☑ : TwinCAT automatically inserts the namespace before the identifier.
Convert keywords to uppercase automatically (AutoFormat)	☑ : TwinCAT automatically writes all keywords in capital letters.
Automatically list selection in cross-reference view	☑ : The Cross Reference List automatically displays the references of the variables/POUs/DUTs that are currently selected or in which the cursor is positioned.

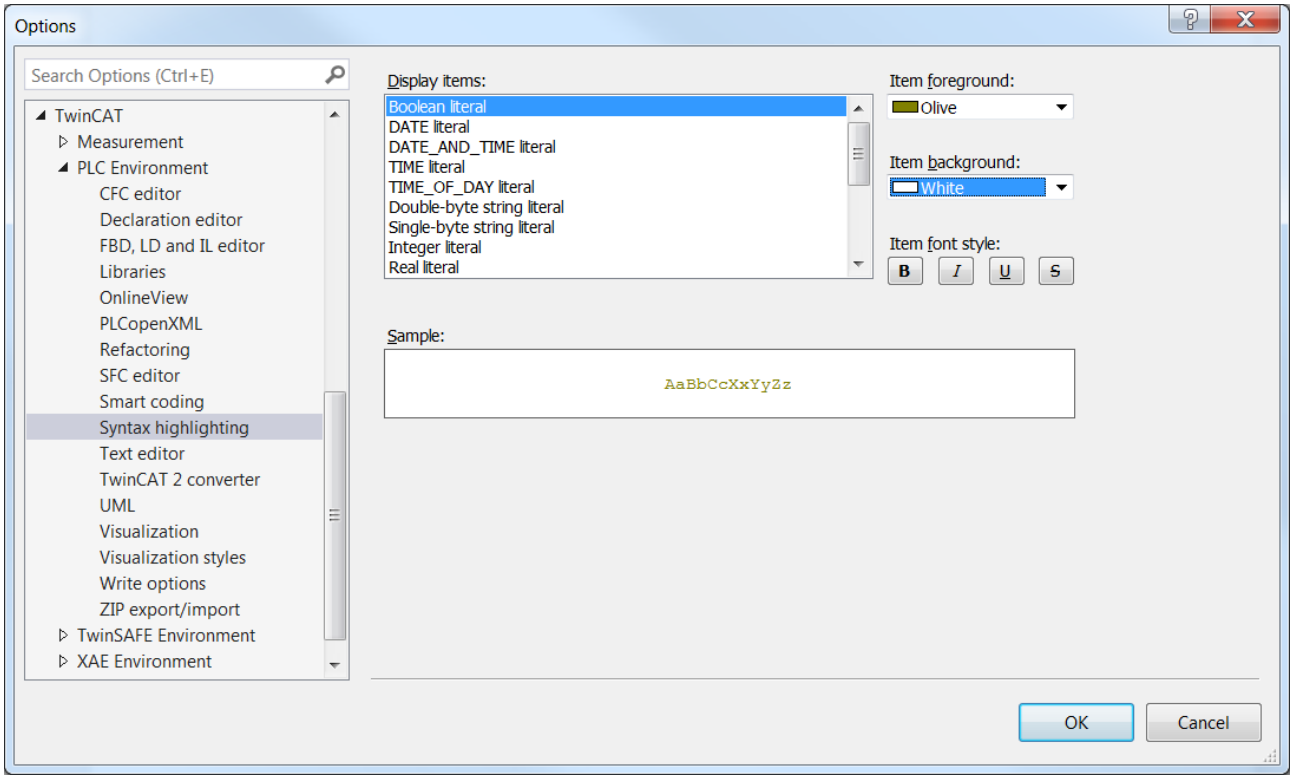
See also:

- PLC documentation: Programming languages and their editors
- PLC documentation: Using the input assistant
- PLC documentation: Find locations where the cross reference list is used

5.9.1.10 Dialog Options - Syntax highlighting

Function: The dialog is used to configure the color and font settings for the text elements of an editor (e. g. operands, pragmas).

Call: TwinCAT > PLC Environment > Syntax highlighting



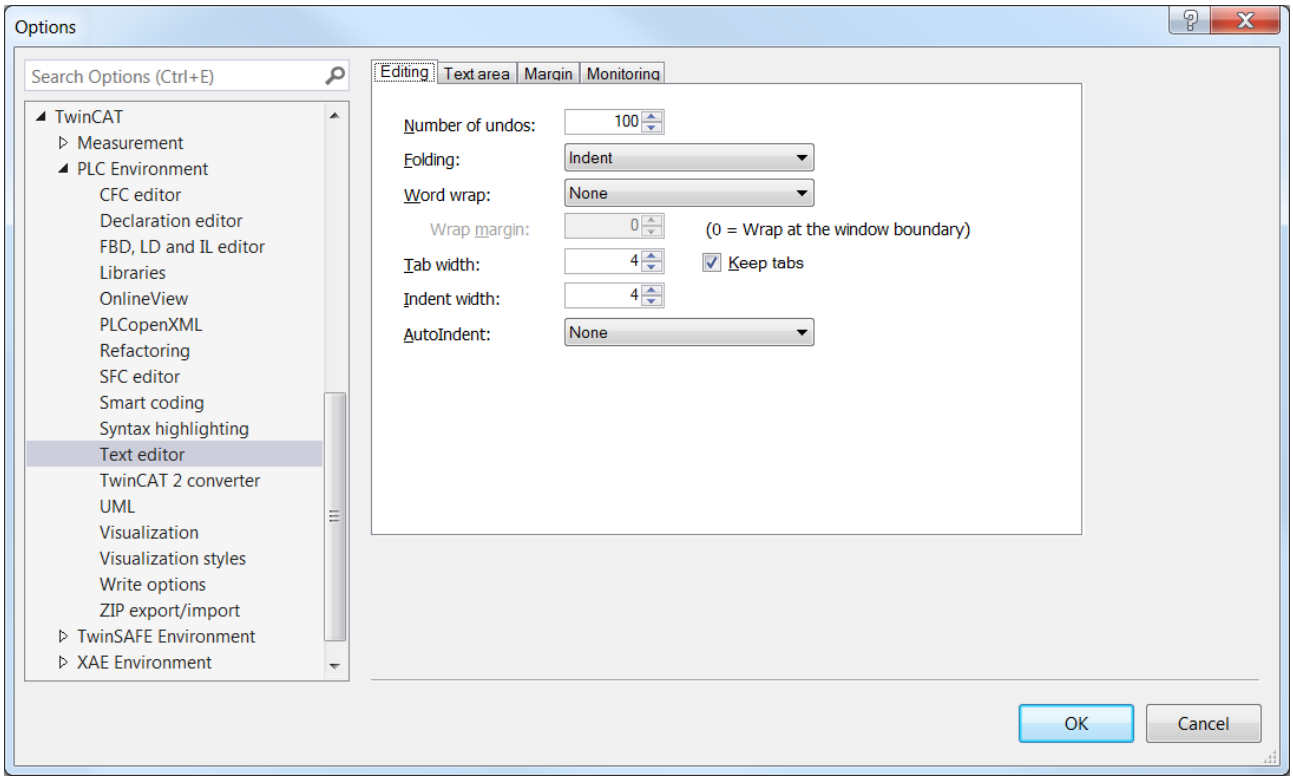
Display items	Selection list for text items
Item foreground	Foreground color of the text item
Item background	Background color of the text item
Item font style	Text item font style (bold, italic, underlined, strikethrough)
Sample	The sample text shows the current settings in a preview

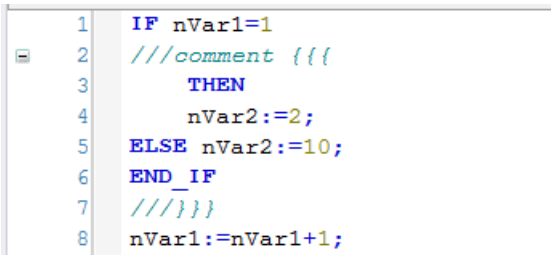
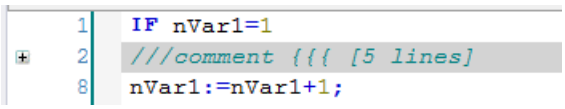
5.9.1.11 Dialog Options - Text editor

Function: The dialog contains settings for working in a text editor.

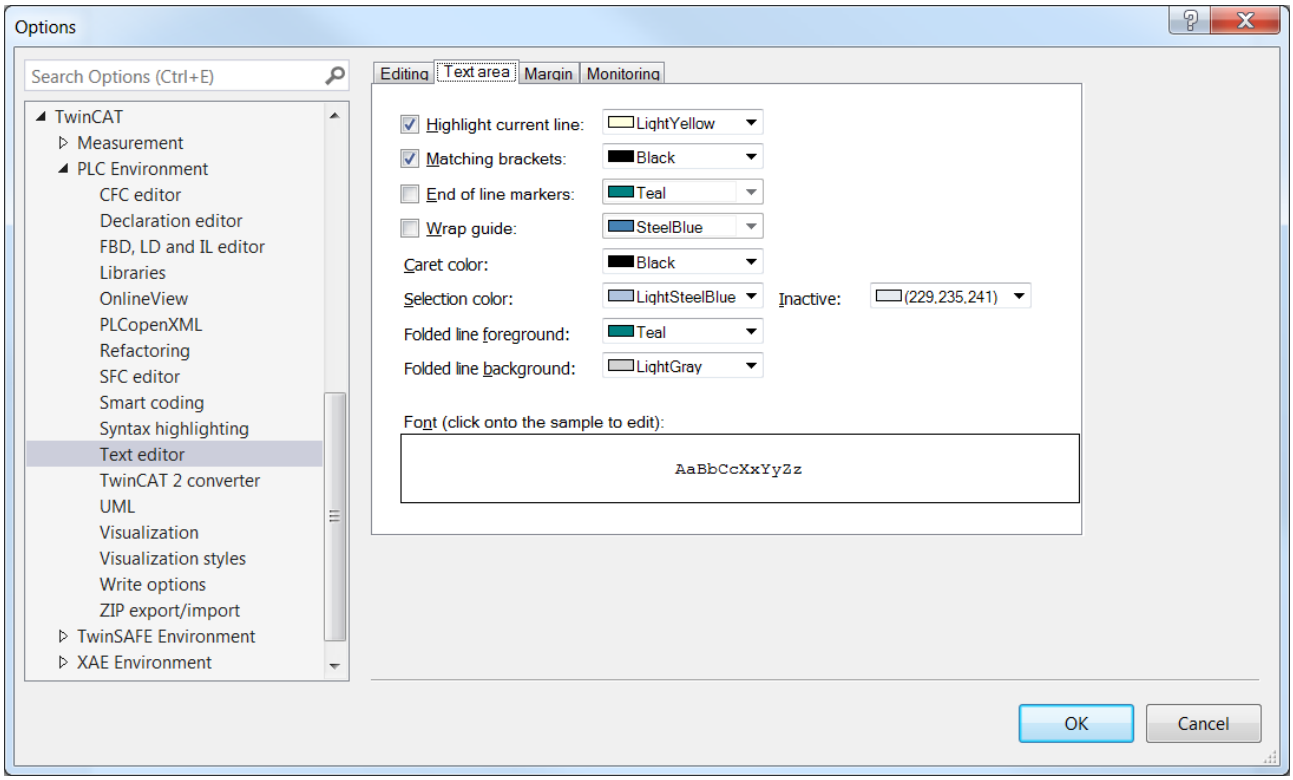
Call: TwinCAT > PLC Environment > Text editor

Editing tab


Options
Search Options (Ctrl+E)
TwinCAT
Measurement
PLC Environment
CFC editor
Declaration editor
FBD, LD and IL editor
Libraries
OnlineView
PLCopenXML
Refactoring
SFC editor
Smart coding
Syntax highlighting
Text editor
TwinCAT 2 converter
UML
Visualization
Visualization styles
Write options
ZIP export/import
TwinSAFE Environment
XAE Environment
Editing
Text area
Margin
Monitoring
Number of undos:
100
Folding:
Indent
Word wrap:
None
Wrap margin:
0
(0 = Wrap at the window boundary)
Tab width:
4
Keep tabs
Indent width:
4
AutoIndent:
None
OK
Cancel

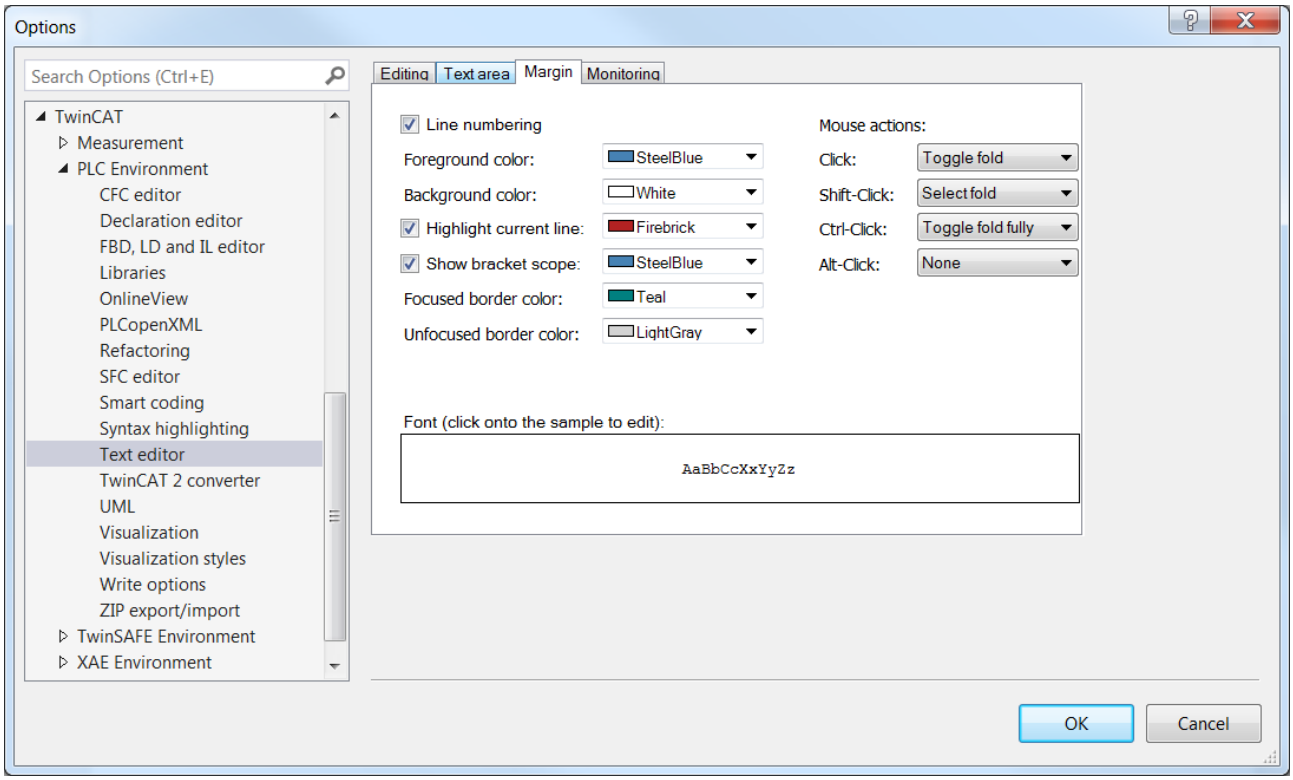
Number of undos	Maximum number of steps for which the command **Edit > Undo** is available.
Folding	Defines the structuring of the code by indentations. When you select an indentation, you can expand or collapse the indentation section using a plus or minus sign to the left of the first line of each section. • Indent: TwinCAT combines all lines that are indented relative to the previous line in one indentation unit. • Explicit: You explicitly use comments to identify the code section to be grouped in an indentation unit: The section must be preceded by a comment containing 3 opening curly brackets "{{{" and followed by a comment containing 3 closing curly brackets "}}}". The comments can contain additional text. Example:  
Word wrap	• Soft: The line break occurs at the edge of the editor window if 0 is entered for the wrap margin. • Hard: The line break occurs after the number of characters specified at the wrap margin.
Tab width	Number of characters
Keep tabs	☑ : The space that you have inserted with the [Tab] key will not subsequently be converted by TwinCAT into individual spaces.
Indent width	If you have enabled the option Smart automatic indentation or Smart with code completion, TwinCAT inserts the corresponding number of blanks at the beginning of the line.
Auto Indent	• Do not indent automatically • Block: A new line automatically adopts the indentation of the previous line. • Intelligent: Lines that follow a line containing a keyword (for example, VAR) automatically indent by the specified indent width. • Smart with code completion: Indentation as with the Smart option, in addition TwinCAT inserts the final keyword (e.g. END_VAR).

Text area tab



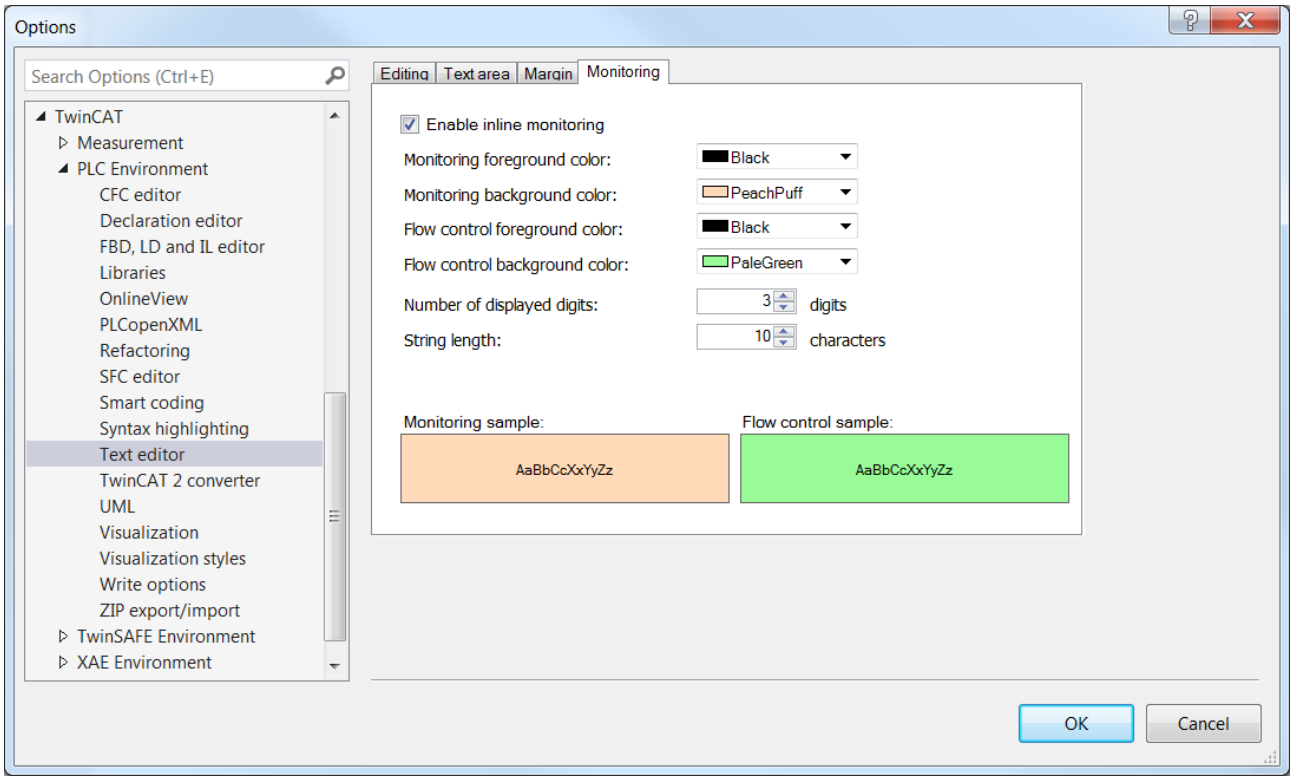
Matching brackets	If the cursor is positioned before or after a bracket within a code line, TwinCAT marks the corresponding closing or opening bracket with a frame in the set color.
Line markers	TwinCAT marks the end of each editor line with a small horizontal line in the specified color after the last character of the line (including spaces).
Wrap guide:	If a soft or hard line break is enabled, the defined wrap guide is displayed by a vertical line in the selected color.
Caret color	Color of the cursor character
Selection color	Color of the selected text area
Inactive	Color of a selection if the corresponding window is not active (focus is on another window).
Folded line foreground	Color of the header line of a closed indented section in the code
Folded line background	Header line of a closed indented section in the code is highlighted in color.
Font	A click on the field opens the standard dialog for configuring the font.

Margin tab



Settings for the left margin of the text editor window, which is separated by a vertical line from the input area:	
Line numbering	Display of the line numbers in the declaration and implementation part, each starting with 1
Foreground color	Line number color
Background color	Margin color
Show bracket scope	Bracketing comprises the lines between the keywords that open and close a construct, such as IF and END_IF. If the option is enabled and the cursor is positioned before, after or in one of the keywords of a construct, the bracketing area is indicated by a square bracket in the margin. You can select the color of the brackets from the selection list for this option.
Focused border color	Color of the dividing line between the margin and the input area
Unfocused border color	Color of the dividing line between the margin and input area of the currently inactive part of the window
Mouse actions	One of the following actions can be assigned to each of the specified mouse actions or mouse button shortcuts. TwinCAT executes the actions when you execute the mouse action on the plus or minus sign before the header line of a bracketed area: • Select fold: TwinCAT selects all the lines of the bracketed area. • Toggle fold: TwinCAT opens or closes the bracketed area, or, in case of nested bracketing, the first level of the bracketed area. • Toggle fold fully: TwinCAT opens or closes all levels of a nested bracketed area.

Monitoring tab



Settings for displaying the monitoring fields	
Enable Inline Monitoring	The monitoring fields are displayed after the variables in online mode
Monitoring foreground color	Representation of the value in the monitoring field
Monitoring background color	Representation of the background in the monitoring field
Flow control foreground color	Representation of the value in the monitoring fields at the flow control items
Flow control background color	Representation of the background of the monitoring fields at the flow control positions
Number of displayed digits	Number of decimal places in the monitoring field
String length	Maximum length of string variable values in the monitoring field

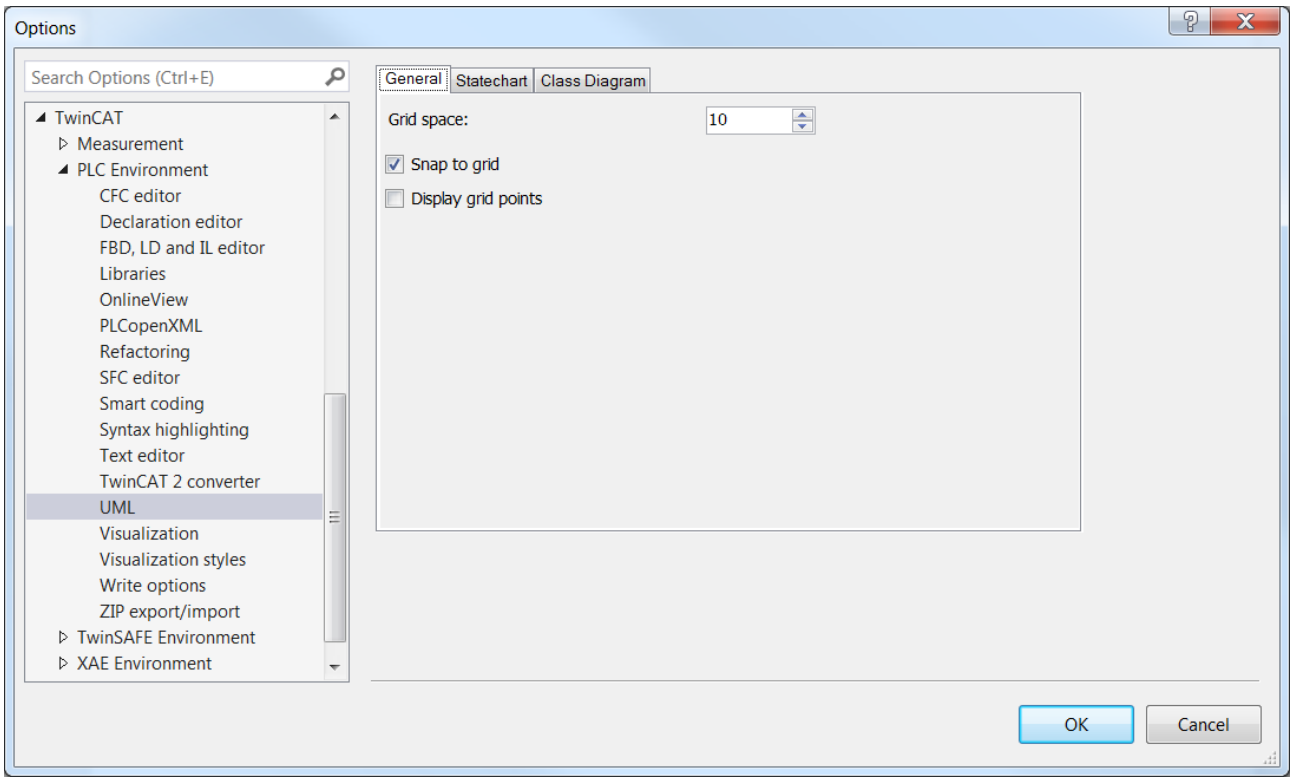
See also:

- PLC documentation: Programming languages and their editors

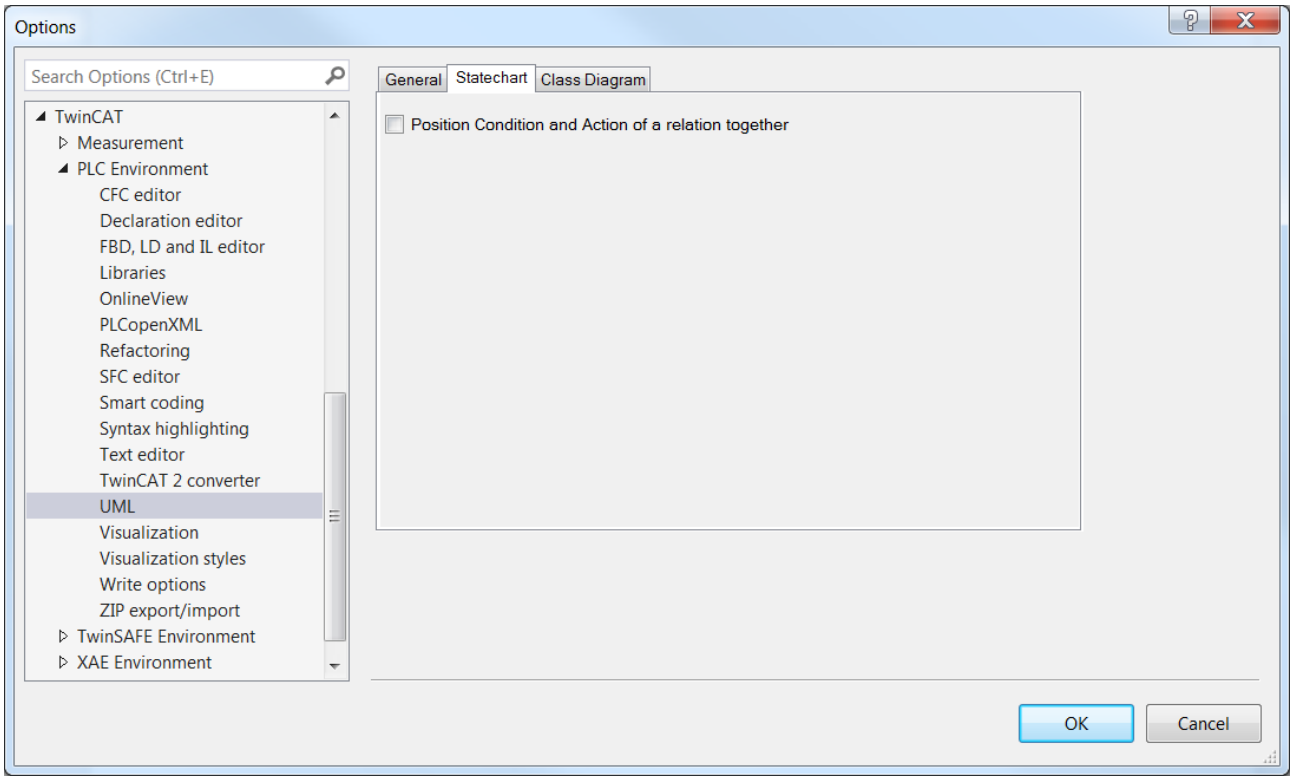
5.9.1.12 Dialog Options - UML

Function: The dialog is used to configure the UML editor.

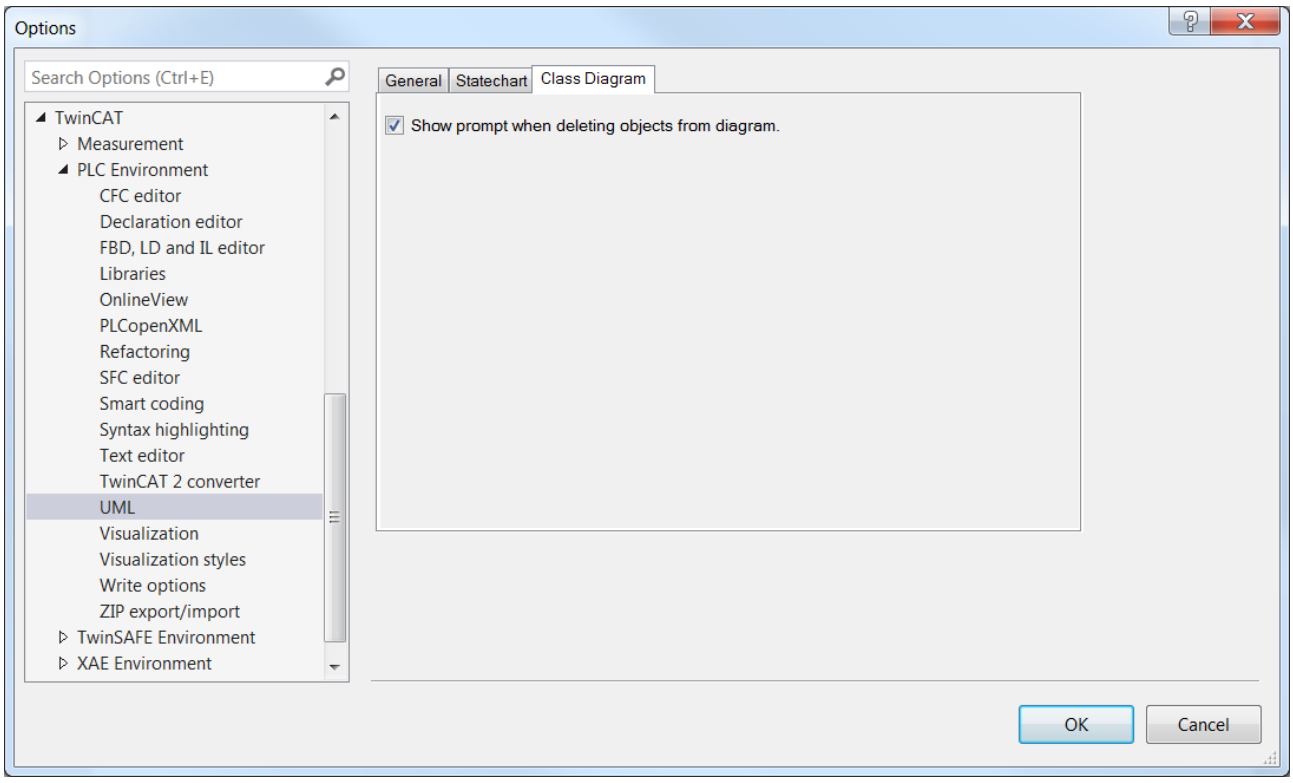
Call: TwinCAT > PLC Environment > UML



Grid space	Gridspace in pixels Default: 10
Snap to grid	☑ All elements in the UML editors are aligned to the grid.
Display grid points	☑ The grid points are displayed in the UML editors.



Position Condition and Action of a relation together	☑ In the Statechart, a guard condition and an action belonging to the same transition are moved synchronously.



Show prompt when deleting objects from diagram	Objects can be deleted either only from the diagram or from the diagram and from the project. ☐ By default, the object is only deleted from the diagram. ☑ When deleting an object, a selection window appears to configure whether the object is to be deleted only from the diagram or from the project.

See also:

- UML documentation: UML

5.9.1.13 Dialog Options - Visualization

Function: The dialog is used to configure the visualization editor.

Call: TwinCAT > PLC Environment > Visualization

General tab

These settings are only used for integrated visualization, i.e. not for the TwinCAT PLC HMI (TargetVisu) and TwinCAT PLC HMI Web display variants.

Presentation options

Fixed	The visualization retains its original size.
Isotropic	The visualization retains its proportions.
Anisotropic	The visualization adapts to the size of the window in the development system
Antialiased Drawing	The visualization is characterized by antialiasing methods, both during editing and as integrated visualization at runtime. Tip: If a horizontal or vertical line is drawn out of focus on a specific visualization platform, this can currently be corrected by a 0.5 px shift in the direction of the line thickness; see item property **Absolute motion**, option **Use REAL values**. Requirement: The platform supports the use of REAL coordinates

Editing options

Linking to shift/key variable	< The placeholder <Shift/Key variable> in the visualization element properties is enabled. If you drag an item with the property Color variables, color change into the visualization editor, this property will be configured with the placeholder <Shift/Key variable>. The following items are affected: button, frame, image, line, circular sector, polygon, rectangle, text field, scrollbar

Grid tab

Grid

Visible	Grid lines are visible in the visualization editor at distance size
Active	Grid lines are active in the visualization editor at distance size. The items are aligned with the grid, all position values are on a grid line. An item that is already in a visualization when the grid is enabled is not automatically aligned. To do this, you must first drag it to another position. The grid lines can be active and invisible
Size	Distance of the grid lines in pixels

File options tab

Text file for textual "List components".	File name and location of a file of type .csv. It contains a table with texts in the format of a text list. The entries in the file are provided if the **List components** function is used as input assistant. You create this file as an export file of the global text list using the Command Import/Export Text Lists.

See also:

- PLC documentation: Creating a visualization

5.9.1.14 Dialog Options - Visualization styles

Function: The dialog is used to configure the visualization styles

Call: TwinCAT > PLC Environment > Visualization styles

Style selection

Display all versions (for experts only)	☐ In addition to the currently selected style, all other styles of the repository are available for selection, but only in the latest version. If newer versions are installed for the selected style, they will also be listed. ☑ All installed styles in all installed versions are available for selection.

Style for new Visualization Managers

Last used: <Style, Version, Manufacturer>	Style that is automatically set as selected when you add a new visualization application. Depending on the device, it is possible that a display variant is displayed in a different way, despite this setting
Preset: <Style, Version, Manufacturer>	
<Style, Version, Manufacturer>	

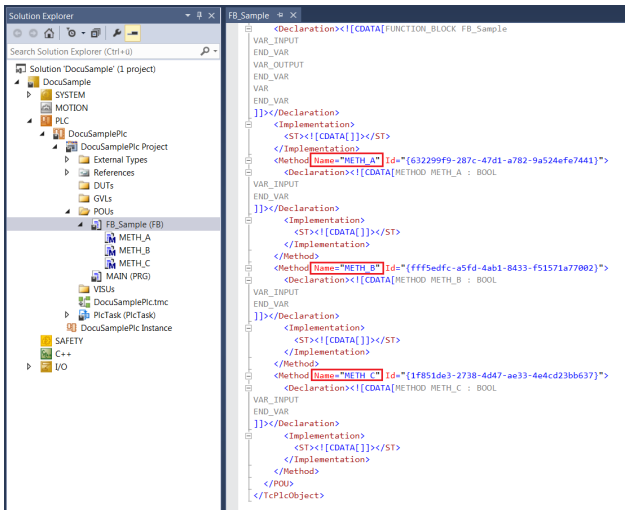
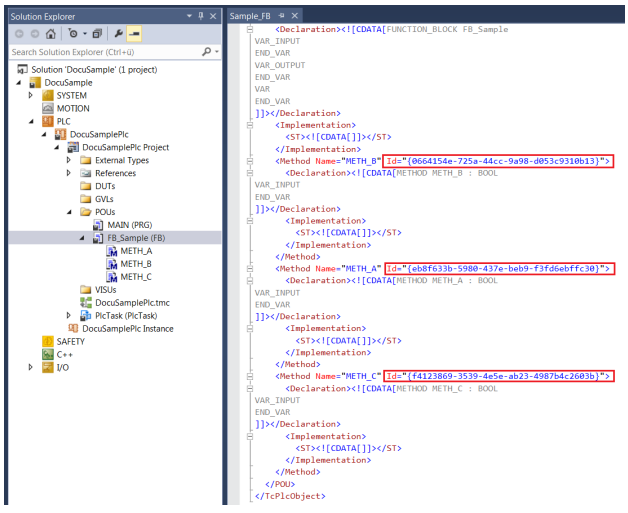
5.9.1.15 Dialog Options - Write Options

Write Options

Separate LineIDs	• TRUE: The line IDs of a POU are stored in a separate file (LineIDs.dbg), so that changes in the line IDs do not lead to changes in the POU, which would then be misinterpreted in the source control system as content changes. Line IDs are required for breakpoint handling, for example, and ensure that the code lines can be assigned to machine code instructions. (Default setting: FALSE)
Sort by name	• TRUE (default setting): The subelements of POUs (actions, methods, properties) are sorted by name and not by internal ID (see sample [▶ 171]).

Sample

The sample illustrates the different storage sequence of the `METH_A`, `METH_B` and `METH_C` methods, depending on whether the **Sort by name** option is enabled or disabled. If the option is disabled (FALSE), the `METH_B` method is not in second place according to its name, but in first place according to its internal ID.

Sort by name = TRUE	Sort by name = FALSE
	

5.9.1.16 Dialog Options - ZIP export/import

Function: The dialog is used to configure the ZIP export and import settings.

Call: TwinCAT > PLC Environment > ZIP export/import

See also:

- PLC documentation: Exporting and importing a PLC project

5.9.2 Command Customize

Function: The command opens the **Customize** dialog. The dialog contains tabs for configuring the user interface. Here you can customize the menus, toolbars and keyboard mapping to suit your individual needs.

Call: Menu **Tools**

You can restore the TwinCAT standard settings at any time via the **Reset** button.

See also:

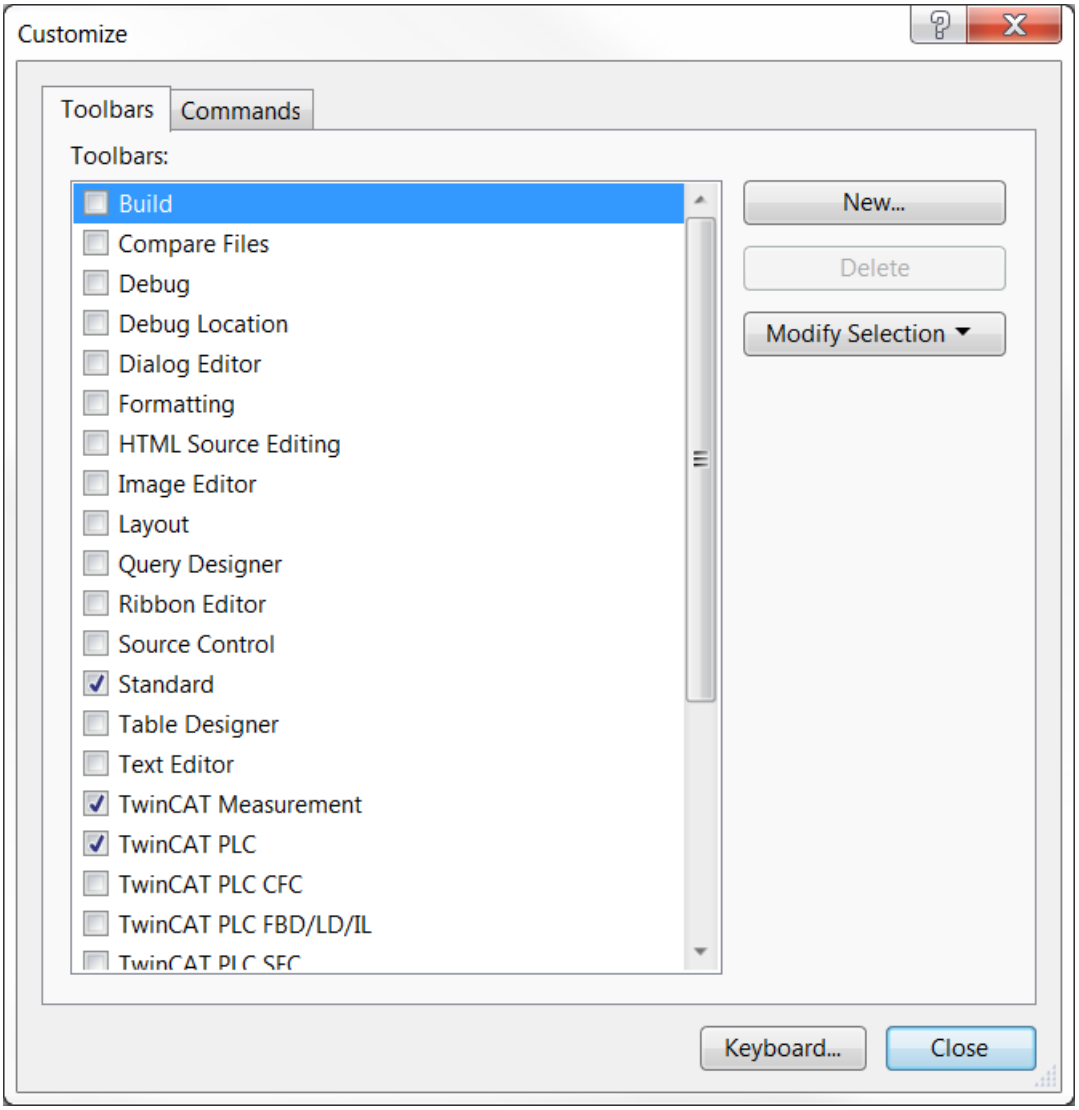
- TC3 User Interface documentation > Customize menus [▶ 34]
- TC3 User Interface documentation > Customize toolbars [▶ 35]
- TC3 User Interface documentation > Customize keyboard shortcuts [▶ 36]

5.9.2.1 Dialog Customize - Toolbars tab

Function: You can use the dialog to create new toolbars or adapt existing toolbars.

Call: Menu **Tools > Customize**

When you close the dialog with **Close**, the changes become visible in the menu bar of the TwinCAT user interface.



New… (Add toolbar)	TwinCAT adds a toolbar above the selected toolbar. A dialog opens, in which a name can be entered.
Delete (remove toolbar)	TwinCAT removes the selected toolbar. You can only remove toolbars you have created yourself.
Modify Selection (position toolbar)	TwinCAT positions the selected toolbar at the top, bottom, left or right frame of the main window
Keyboard....	Opens the **Options** dialog, in which you can define keyboard shortcuts.

Toolbars

Displays the currently defined toolbars.	
☐ (hide)	Hides the selected toolbar on the user interface.
☑ (show)	Shows the selected hidden toolbar in the TwinCAT user interface.

See also:

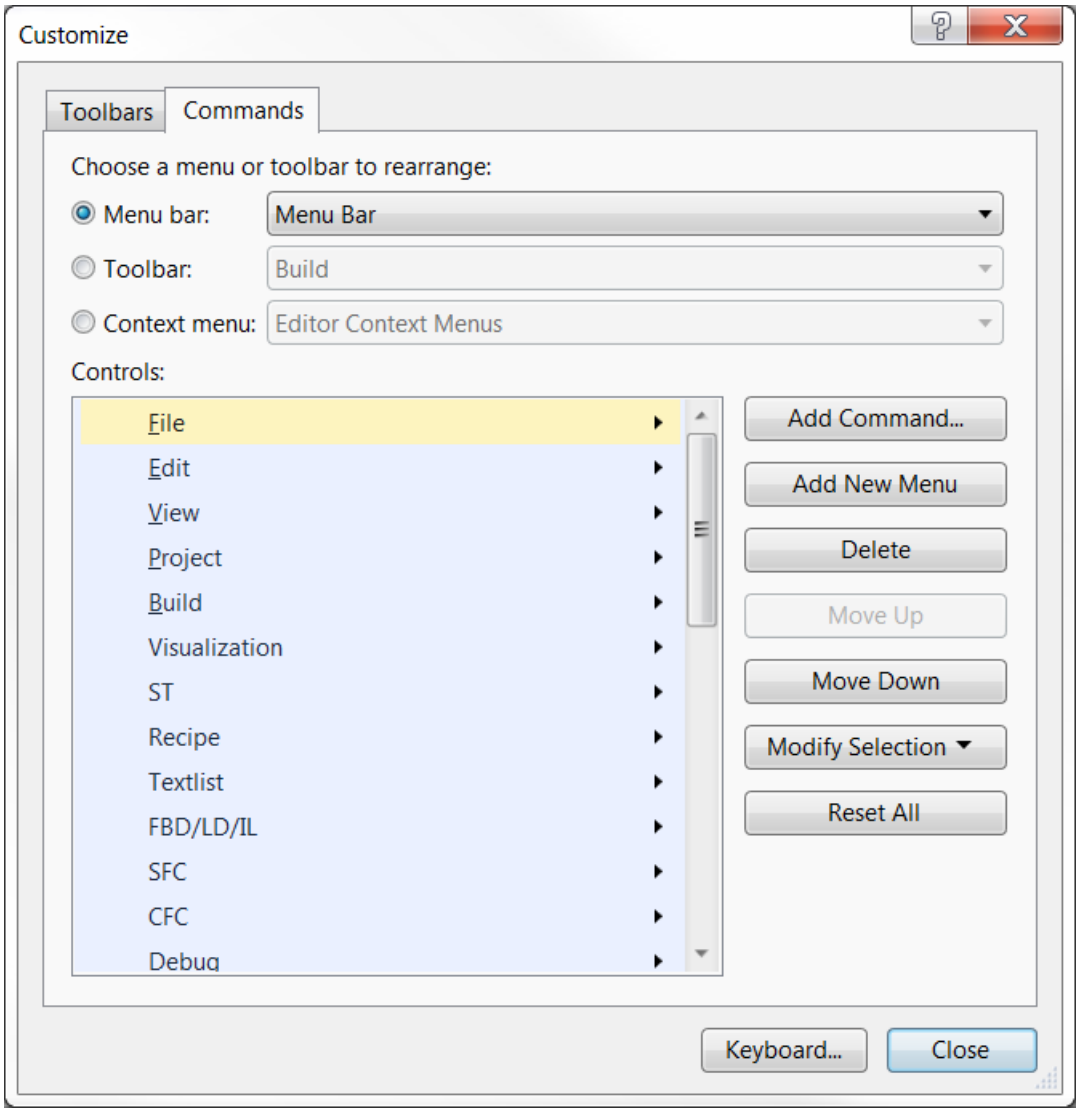
- TC3 User Interface documentation > Customize toolbars [▶ 35]

5.9.2.2 Dialog Customize - Commands tab

Function: The dialog allows you to define commands, as well as the structure and content of the menus and toolbars for the user interface.

Call: Menu **Tools > Customize**

When you close the dialog with **Close**, the changes become visible in the menu bar of the TwinCAT user interface.



Menus and Toolbars

Display of the currently defined toolbars, menus, submenus and the commands contained therein.	
Menu bar	List of menus and submenus
Toolbar	List of toolbars
Context menu	List of context menus

Controls

Controls	Listing of commands or submenus contained in the selected menu or toolbar. The arrangement from top to bottom corresponds to the arrangement shown later in the TwinCAT menu or in the toolbar.

Add Command	Opens the **Add Command** dialog. The **Add Command** dialog is used to select one or more commands. Left part: List of categories. Right part: Listing of commands of the selected category. Adds a command above the selected command.
Add New Menu	Adds a new menu above the selected menu.
Modify Selection	Opens a menu in which the name of a newly added menu can be determined.
Delete	Removes the selected menu or command.
Move Up	Moves the selected command or menu up in the order of commands or menus.
Move Down	Moves the selected command or menu down in the order of commands or menus.
Reset All	Resets the entire menu to the default settings.
Keyboard	Opens the **Options** dialog, in which you can define keyboard shortcuts.

5.10 Window

5.10.1 Command Float

Function: The command detaches a view or window that is docked (fixed) to the frame of the user interface from the frame and places it on the screen as a detached window.

Call: Window menu, context menu or button in the header of the view or the tab (window)

The view can then also be placed outside the user interface. Use the **Dock** command to reattach a detached view to the user interface frame.

See also:

- Command Dock [▶ 175]
- TC3 User Interface documentation: Arranging views and windows [▶ 37]

5.10.2 Command Dock

Function: The command "docks" a view, which was previously detached with the **Float** command and is now positioned on the screen as a detached view, back to the user interface frame.

Call: Window menu, context menu or button in the header of the view or the tab (window)

See also:

- Command Float [▶ 175]
- TC3 User Interface documentation: Arranging views and windows [▶ 37]

5.10.3 Command Hide

Symbol: 

Function: The command hides a view.

Call: Window menu, context menu or button in the header of the view

Requirement: A view is enabled.

Hide means that the view is closed. The command thus corresponds to closing a view via the  button in the header of the view. Use the commands on the **View** menu to unhide or open a hidden view or open.

See also:

- View [▶ 78]
- TC3 User Interface documentation: Show/hide views [▶ 38]

5.10.4 Command Auto Hide All

Function: The command hides all views.

Call: Window menu, context menu or button in the header of the view

Hiding means that TwinCAT displays all views only as a tab within the user interface and that it only

becomes visible when you click on the tabs. If you then click the  button in the header bar of the view or select the **Dock** command, the view will be reattached to the user interface.

See also:

- Command Dock [▶ 175]
- View [▶ 78]
- TC3 User Interface documentation: Show/hide views [▶ 38]

5.10.5 Command Auto Hide

Symbol: 

Function: The command puts a view into the background.

Call: Window menu, context menu or button in the header of the view

Requirement: A view is enabled.

Putting in the background means that TwinCAT only displays the view as a tab within the user interface,

which only becomes visible when you click on the tab. If you then click the  button in the header bar again or select the **Dock** command, the view will be reattached to the user interface.

See also:

- Command Dock [▶ 175]
- TC3 User Interface documentation: Show/hide views [▶ 38]

5.10.6 Command Pin tab

Symbol: 

Function: The command attaches the currently active tab to the left margin of the main window.

Call: Window menu, context menu or button in the header bar of the tab (window)

Requirement: A tab (window) is enabled.

See also:

- TC3 User Interface documentation: Show/hide views [▶ 38]

5.10.7 Command New Horizontal Tab Group

Symbol: 

Function: The command moves the active window to a new, separate tab group below the existing one.

Call: Window menu, context menu of the header bar of the tab (window)

Requirement: Several editor windows are arranged next to each other as tabs.

If you open another object in the editor, this is automatically placed in the tab group where the focus is.

See also:

- TC3 User Interface documentation: Arranging views and windows [▶ 37]
- Command New Vertical Tab Group [▶ 177]

5.10.8 Command New Vertical Tab Group

Symbol: 

Function: The command moves the active window to a new, separate tab group to the right of the existing one.

Call: Window menu, context menu of the header bar of the tab (window)

Requirement: Several editor windows are arranged next to each other as tabs.

If you open another object in the editor, this is automatically placed in the tab group where the focus is.

See also:

- TC3 User Interface documentation: Arranging views and windows [▶ 37]
- Command New Horizontal Tab Group [▶ 177]

5.10.9 Command Reset Window-Layout

Function: This command resets all currently open windows and views to their default positions. You need to confirm the command before it is executed.

Call: Window menu

5.10.10 Command Close All Documents

Symbol: 

Function: The command closes all currently open editor windows.

Call: Window menu

Requirement: At least one editor window is open.

See also:

- TC3 User Interface documentation: Show/hide views [▶ 38]

5.10.11 Command Window

Function: The command opens the **Window** dialog, which displays all open objects. You can activate or close windows in it.

Call: Window menu

5.10.12 Window submenu commands

Function: The command activates the selected window.

Call: Window menu

For each open editor window the **Window** menu contains a command **<n><Object name>**, through which you can activate the window, i.e. put the focus on it. In offline mode, TwinCAT adds the extension (offline) after the command. For function blocks, the extension (impl) or <instance path> is added to distinguish between implementation and instance.

See also:

- Command Window [▶ 178]

5.11 SFC

5.11.1 Command Init step

Symbol: 

Function: The command converts the currently selected step to an init step.

Call: Menu **SFC**, context menu

Executing the command changes the frame of the step element to a double line. The step, which previously was the init step, automatically becomes a 'normal' step and is represented with single frame.

The property **Init step** can also be enabled or disabled in the **Properties** view of a step element, although in this case TwinCAT does not automatically adjust the settings of the other steps.

The command can be useful if you want to change the chart. When you create a new SFC object, it automatically includes an init step, followed by a transition (TRUE) and a jump back to the init step.

Note the option to reset the chart to the init step by using the SFC flags SFCInit and SFCReset in online mode.

See also:

- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: SFC element properties

5.11.2 Command Insert step transition

Symbol: 

Function: The command adds a step and a transition before the currently selected position.

Call: Menu **SFC**, context menu

If you have selected a step, TwinCAT inserts a new step transition combination. If you have selected a transition, a new transition step combination is inserted.

The new step is called step<n> by default. n is a sequential number, starting with 0 for the first step, which is added in addition to the init step. Accordingly, the new transition is called Trans<n> by default. You can edit the default names directly by clicking on the name.

See also:

- Command Insert step-transition after [▶ 179]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: SFC elements step and transition

5.11.3 Command Insert step-transition after

Symbol: 

Function: The command adds a step and a transition after the currently selected position.

Call: Menu **SFC**, context menu

If you have selected a step, TwinCAT inserts a new transition step combination. If you have selected a transition, a new step transition combination is inserted.

The new step is called Step<n> by default. n is a sequential number, starting with 0 for the first step, which is added in addition to the init step. Accordingly, the new transition is called Trans<n> by default. You can edit the default names directly by clicking on the name.

See also:

- Command Insert step transition [▶ 178]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: SFC Elements 'Step' and 'Transition'

5.11.4 Command Parallel

Symbol: 

Function: The command converts the selected alternative branch to a parallel branch.

Call: Menu **SFC**, context menu

Requirement: The horizontal connecting line of a branch is selected.

Note that after converting a branch, you must check and adjust the sequence of steps and transitions before and after the branch.

See also:

- Command Alternative [▶ 180]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor

5.11.5 Command Alternative

Symbol: 

Function: The command converts the selected parallel branch to an alternative branch.

Call: Menu **SFC**, context menu

Requirement: The horizontal connecting line of a branch is selected.

Note that after converting a branch, you must check and adjust the sequence of steps and transitions before and after the branch.

See also:

- Command Parallel [▶ 179]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor

5.11.6 Command Insert Branch

Symbol: 

Function: The command adds a branch to the left of the currently selected position.

Call: Menu **SFC**, context menu

The behavior of the command corresponds to the command **Insert branch right**.

See also:

- Command Insert branch right [▶ 180]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: SFC element Branch

5.11.7 Command Insert branch right

Symbol: 

Function: The command adds a branch to the right of the currently selected position.

Call: Menu **SFC**, context menu

The type of the inserted branch depends on the selected element:

- If the top-most element of the currently selected elements is a transition or an alternative branch, TwinCAT inserts an alternative branch.
- If the top-most element of the currently selected element is a step, a macro, a jump or a parallel branch, TwinCAT adds a parallel branch with label Branch<x>, where x is a sequential number. You can edit this default label name. You can specify the label as the target of a jump.
- If a common element of an existing branch is currently selected (horizontal line), TwinCAT adds the new branch at the far right as an additional branch. If an entire branch of an existing branch is currently selected, TwinCAT adds the new branch directly to the right of it as a new branch.

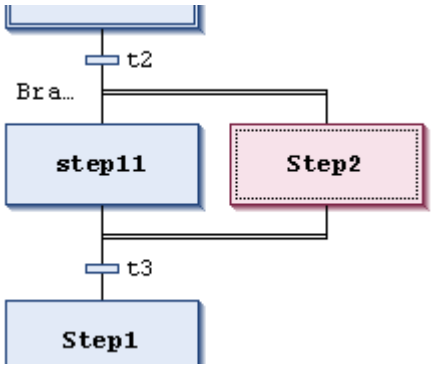
Note that you can use the commands **Alternative** or **Parallel** to convert a branch to the respective other type.

Example: Parallel branch

The following diagram shows a newly inserted parallel branch, generated with Command Insert branch right, while step11 was selected. TwinCAT automatically inserts a step (Step2 in the example).

Processing in online mode: If t2 returns TRUE, TwinCAT executes Step2 immediately after step11, before t3 is evaluated.

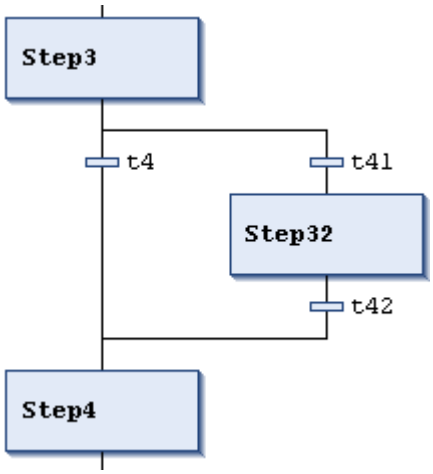
In contrast to alternative branches, TwinCAT therefore executes both branches.



Example: Alternative branch

The following diagram shows a newly inserted alternative branch, generated with Command Insert branch right, while transition t4 was selected. TwinCAT automatically inserts a step (Step32 in the example), a preceding and a subsequent transition (t41, t42).

Processing in online mode: If Step3 is active, TwinCAT evaluates the following transitions (t4, t41) from left to right. The first junction of the branch in which the first transition returns TRUE is executed. In contrast to parallel branching, only one junction is executed.



See also:

- Command Alternative [▶ 180]
- Command Parallel [▶ 179]
- Command Insert Branch [▶ 180]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: SFC element Branch

5.11.8 Command Insert action association

Symbol:

Function: The command assigns an IEC action to a step.

Call: Menu **SFC**, context menu

Requirement: A step is selected.

TwinCAT adds the action element to the right of the currently selected step element.

If you have already assigned one or more actions to the step, they are displayed in a "action list", one below the other. The new action is then placed as follows:

- As the first action of the step, i.e. at the top of the action list, if you have selected the step element.
- Directly before, i.e. above the action, if you have selected one of the existing actions in the action list of the step.

The left part of the action element contains the qualifier, by default N, in the right part you enter the action name. To do this, click in the box to get an editing frame. You must have created this action as a POU in the project.

You can also edit the qualifier. A list of the valid qualifiers is described in section "Qualifiers for actions in SFC".

See also:

- Command Insert action association after [▶ 182]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: Qualifier for actions in SFC

5.11.9 Command Insert action association after

Symbol: 

Function: The command assigns an IEC action to a step.

Call: Menu **SFC**, context menu

Requirement: A step is selected.

The command corresponds to the description of Insert action association. The difference is that TwinCAT places the new action not at the first, but at the last position of the action list. If you have selected an action in the action list, TwinCAT does not place the new action above it, but below it.

See also:

- Command Insert action association [▶ 181]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: Qualifier for actions in SFC

5.11.10 Command Insert jump

Symbol: 

Function: The command inserts a jump element before the currently selected element.

Call: Menu **SFC**, context menu

Requirement: A step is selected.

TwinCAT automatically inserts the jump with jump target step. You must then replace this jump target with a real jump target. You can select the target with the Input Assistant.

See also:

- Command Insert jump after [▶ 183]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: SFC element Jump

5.11.11 Command Insert jump after

Symbol:

Function: The command inserts a jump element after the currently selected element.

Call: SFC menu

TwinCAT automatically inserts the jump with jump target step. You must then replace this jump target with a real jump target. You can select the target with the Input Assistant.

See also:

- Command Insert jump [▶ 182]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: SFC element Jump

5.11.12 Command Insert macro

Symbol:

Function: The command inserts a macro element before the currently selected element.

Call: Menu **SFC**, context menu

The new macro is called Macro<x> by default. x is a sequential number, starting with 0 for the first macro. You can edit the default name directly by clicking on the name.

To edit the macro, open it with the command **Show macro** in the macro editor.

See also:

- Command Show macro [▶ 184]
- Command Insert macro after [▶ 183]
- PLC documentation: Sequential Function Chart (SFC)
- SFC Editor

5.11.13 Command Insert macro after

Symbol:

Function: The command inserts a macro element after the currently selected element.

Call: Menu **SFC**, context menu

The command corresponds to the description of the command **Insert macro**.

See also:

- Command Insert macro [▶ 183]
- Command Show macro [▶ 184]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor

5.11.14 Command Show macro

Symbol: 

Function: The command opens a macro in the macro editor for editing.

Call: SFC menu

Requirement: A macro is selected.

The command causes TwinCAT to close the main view of the SFC editor and open the macro editor instead. This is also an SFC editor, in which you can now edit the part of the SFC diagram that is displayed as a macro box in the main view.

Use the command **Exit macro** to return to the main view.

See also:

- Command Exit macro [▶ 184]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor

5.11.15 Command Exit macro

Symbol: 

Function: The command closes the macro editor and returns to the main view of the SFC editor.

Call: SFC menu

Requirement: A macro is open in the macro editor.

See also:

- Command Show macro [▶ 184]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor

5.11.16 Command Insert after

Symbol: 

Function: The command inserts the elements from the clipboard after the currently selected position.

Call: SFC menu

5.11.17 Command Add entry action

Symbol: 

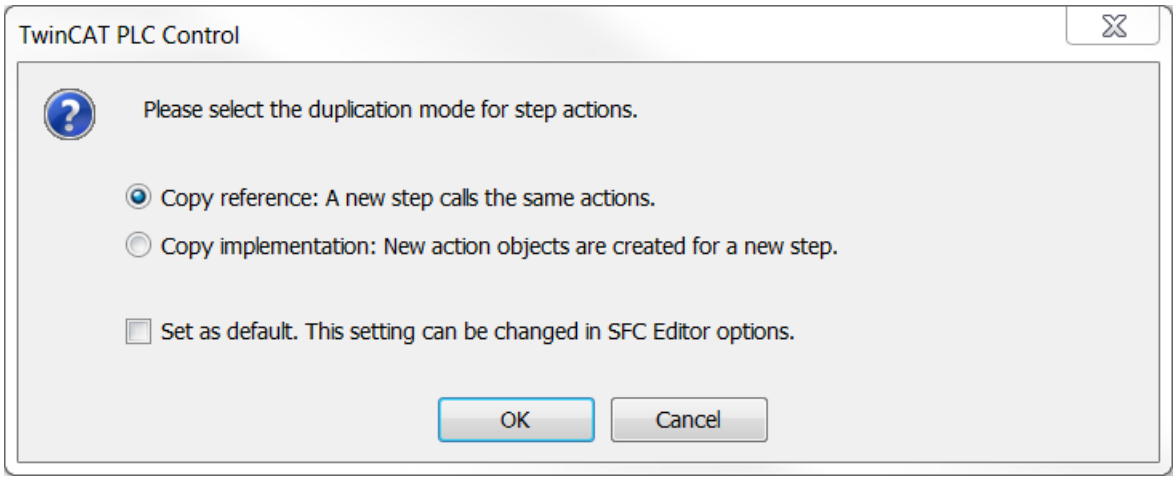
Function: The command leads to the **Add entry action** dialog, in which you define a new step action of type "entry action". Depending on the SFC options, a prompt for selecting the duplication mode for the new step action may appear beforehand.

Call: SFC menu, context menu of the selected step element

Requirement: A step element is selected.

The entry action is automatically opened in the ST editor. The step element is assigned an "E" in the lower left corner.

Query dialog for selecting the duplication mode



Copy reference. A new step will call the same actions	If the step is copied in the SFC, the link to the step action(s) is also copied. The steps copied from one another will therefore call all the same actions.
Copy implementation. New action objects are created for the new step	This means "embedding" of the step actions for the copied steps. By default, the newly created action objects appear below the SFC function block in the PLC project tree in the Solution Explorer. Initially, these objects contain a copy of the original implementation code for the respective action.
Set as default. This setting can be changed in the SFC editor options.	The settings in the dialog are accepted as the default setting. You can change the default setting in the TwinCAT options in the category **SFC editor**. To do this, in the group field **Step actions** in the drop-down list **Standard insert method** select the entry **Always ask**, **Copy reference** or **Duplicate implementation.**

See also:

- Command Options > Dialog Options - SFC editor [▶ 157]
- PLC documentation: Command Add exit action [▶ 186]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: Programming in Sequential Function Chart (SFC)
- PLC documentation: SFC element Action

5.11.18 Command Add exit action

Symbol: 

Function: The command leads to the **Add exit action** dialog, in which you define a new step action of type entry action. Depending on the SFC options, a prompt for selecting the duplication mode for the new step action may appear beforehand. Please refer to the help page for the command **Add entry action**.

Call: SFC menu, context menu of the selected step element

Requirement: A step element in the SFC is selected.

See also:

- Command Add entry action [▶ 185]
- Command Options > Dialog Options - SFC editor [▶ 157]
- PLC documentation: Sequential Function Chart (SFC)
- PLC documentation: Programming in Sequential Function Chart (SFC)
- PLC documentation: SFC Editor
- PLC documentation: SFC element Action

5.11.19 Command Change duplication - Set

Function: The command permanently links each step action or transition that is called by a step or a transition in the SFC function block to the caller. Thus the action or transition object can then only be called by exactly this one caller (pseudo-embedding). As a result, copying step and transition elements that call actions or transitions automatically creates new action or transition objects. The implementation code is copied in each case.

Call: SFC menu

For details of the duplication mode, see the help page for the SFC element properties and the instructions for adding step actions.

See also:

- PLC documentation: SFC Element Properties
- PLC documentation: Programming in Sequential Function Chart (SFC)

5.11.20 Command Change duplication - Remove

Function: The command removes the fixed link of action or transition objects with the step or transition that it calls for the entire SFC function block. This cancels the pseudo-embedding of the action or transition objects. If step and transition elements that call actions or transitions are then copied, the copies call the same actions and transitions as the source.

Call: SFC menu

For details of the duplication mode, see the help page for the SFC element properties and the instructions for adding step actions.

See also:

- PLC documentation: SFC Element Properties
- PLC documentation: Programming in Sequential Function Chart (SFC)

5.11.21 Command Insert step

The command is not included as standard in the **SFC** menu.

Symbol:

Function: The command adds a step before the currently selected position.

Call: SFC menu, Context menu in the SFC editor

The new step is called Step<n> by default. n is a sequential number, starting with 0 for the first step, which is added in addition to the init step. The name can be edited by clicking on it.

Inserting a step without a transition or a transition without a step results in a compile error.

See also:

- Command Insert step-transition after [▶ 179]
- Command Init step [▶ 178]
- PLC documentation: SFC elements step and transition

5.11.22 Command Insert step after

The command is not included as standard in the **SFC** menu.

Symbol:

Function: The command inserts a step after the currently selected position.

Call: SFC menu, Context menu in the SFC editor

The new step is called Step<n> by default. n is a sequential number, starting with 0 for the first step, which is added in addition to the init step. The name can be edited by clicking on it.

Inserting a step without a transition or a transition without a step results in a compile error.

See also:

- Command Init step [▶ 178]
- Command Insert step-transition after [▶ 179]
- PLC documentation: SFC element Jump

5.11.23 Command Insert transition

The command is not included as standard in the **SFC** menu.

Symbol:

Function: The command adds a transition before the currently selected position.

Call: SFC menu, Context menu in the SFC editor

The new transition is called Trans<n> by default. n is a sequential number, starting with 0 for the first transition. The name can be edited by clicking on it.

Inserting a step without a transition or a transition without a step results in a compile error.

See also:

- Command Insert step-transition after [▶ 179]
- PLC documentation: SFC elements step and transition

5.11.24 Command Insert transition after

The command is not included as standard in the **SFC** menu.

Symbol: 

Function: The command inserts a transition after the currently selected position.

Call: SFC menu, Context menu in the SFC editor

The new transition is called Trans<n> by default. n is a sequential number, starting with 0 for the first transition. The name can be edited by clicking on it.

Inserting a step without a transition or a transition without a step results in a compile error.

See also:

- Command Insert step after [▶ 187]
- PLC documentation: SFC elements step and transition

5.12 CFC

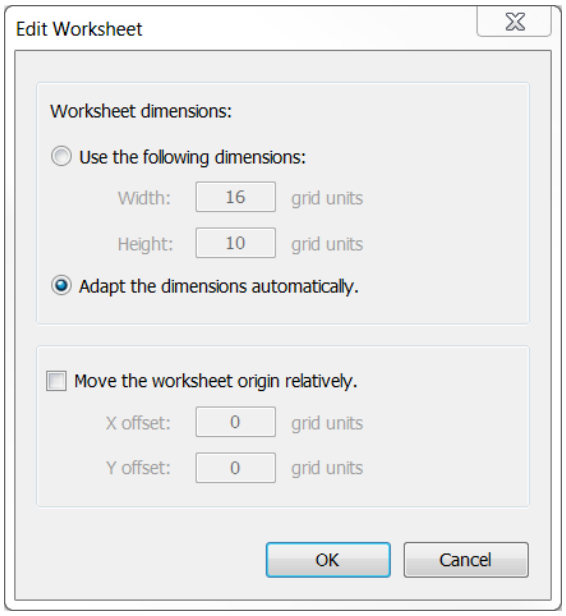
5.12.1 Command Edit Worksheet

Function: The command opens the **Edit Worksheet** dialog, in which you specify the size of the worksheet.

Call: CFC menu

Requirements: A CFC editor is active.

Edit Worksheet dialog



Use the following dimensions	Here, you set the size of the worksheet. Your change is only be accepted if the size is sufficient for the existing program.
Adapt the dimensions automatically	Automatically adapts the size of the worksheet to the size of your program.
Move the worksheet origin relatively	Moves the worksheet on the x or y axis. Entering negative numbers is allowed.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.2 Command Negate

Symbol: 

Function: The command negates the selected function block input or output.

Call: CFC menu, context menu

Requirements: A CFC editor is active. A function block input or output is selected.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.3 Command EN/ENO

Symbol: 

Function: The command adds a Boolean input EN (Enable) and a Boolean output ENO (Enable Out) to the selected function block.

Call: CFC menu, context menu

Requirements: A CFC editor is active. A function block is selected.

The added input “EN” enables the function block. The function block is only executed if it has the value TRUE. The value of this signal is output at output ENO.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.4 Command None

Symbol: 

Function: The command removes a reset (R) or set (S) from the input of the "output" element“.

Call: Menu **CFC > Set/Reset**, context menu > Set/Reset

Requirements: A CFC editor is active. The input of an **Output** element is selected.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.5 Command R-Reset

Symbol: 

Function: The command adds a reset to the input of a Boolean element **Output**.

Call: Menu **CFC > Set Reset**, context menu > Set/Reset

Requirements: A CFC editor is active. The input of an "Output" element is selected.

If an **Output** element has a reset input, the Boolean output value is set to FALSE when the value of the input is TRUE. The value FALSE at the output is retained, even if the input value changes again.

See also:

- Command S-SET [▶ 190]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.6 Command S-SET

Symbol: 

Function: The command adds a set (S) to the input of a Boolean element "Output".

Call: Menu **CFC > Set/Reset**, context menu > Set/Reset

Requirements: A CFC editor is active. The input of an **Output** element is selected.

If an **Output** element has a set input, the Boolean output value is set to TRUE when the value of the input is TRUE. The value TRUE at the output is retained, even if the input value changes again.

See also:

- Command R-Reset [▶ 190]

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.7 Command REF = (reference assignment)

Symbol: 

Function: The command assigns a reference to an "Output" element.

Call: Menu **CFC > Set/Reset**, context menu > Set/Reset

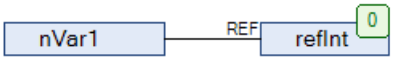
Requirements: A CFC editor is active. The input of an **Output** element is selected.

Example:

Declaration:

```
refInt : REFERENCE TO INT;
nVar1 : INT;
```

CFC:



This corresponds to the ST code

```
refInt REF= nVar1;
```

Further information can be found in the description of the REFERENCE TO data type.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor
- PLC documentation: Reference

5.12.8 Command Move to Beginning

Symbol: 

Function: The command places the selected elements at the beginning of the execution order.

Call: Menu **CFC > Execution Order**, context menu > Execution Order

Requirements: A CFC editor is active. At least one element is selected.

The command only changes the chronological order in the program sequence. The position of the elements is not changed. If you apply the command to more than one element at a time, the sequence is retained within the selected elements.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.9 Command Move to End

Symbol: 

Function: The command places the selected elements at the end of the execution order.

Call: Menu **CFC > Execution Order**, context menu > Execution Order

Requirements: A CFC editor is active. At least one element is selected.

The command only changes the chronological order in the program sequence. The position of the elements is not changed. If you apply the command to more than one element at a time, the sequence is retained within the selected elements.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.10 Command Forward by one

Symbol: 

Function: The command moves the selected elements forward by one in the execution order.

Call: Menu **CFC > Execution Order**, context menu > Execution Order

Requirements: A CFC editor is active. At least one element is selected.

The command only changes the chronological order in the program sequence. The position of the elements is not changed. If you apply the command to more than one element at a time, the sequence is retained within the selected elements.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.11 Command Back by one

Symbol: 

Function: The command moves the selected elements back by one in the execution order.

Call: Menu **CFC > Execution Order**, context menu > Execution Order

Requirements: A CFC editor is active. At least one element is selected.

The command only changes the chronological order in the program sequence. The position of the elements is not changed. If you apply the command to more than one element at a time, the sequence is retained within the selected elements.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.12 Command Order by Data Flow

Function: The command arranges the execution order of the elements in the program according to their position in the network.

Call: Menu **CFC > Execution Order**, context menu > Execution Order

Requirements: A CFC editor is active.

The command only changes the chronological order in the program sequence. The position of the elements is not changed. The command assigns a new number to all elements in the program, even if not all elements are selected when the command is executed.

See also:

- Command Order By Topology [▶ 193]
- Command Set Execution Order [▶ 193]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.13 Command Order By Topology

Function: The command arranges the execution order of the elements according to their position from right to left.

Call: Menu **CFC > Execution Order**, context menu > Execution Order

Requirements: A CFC editor is active. One element is selected.

The command only changes the chronological order in the program sequence. The position of the elements is not changed. The command affects all elements in the program, even if not all elements are selected when the command is executed.

See also:

- Command Order by Data Flow [▶ 192]
- Command Set Execution Order [▶ 193]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.14 Command Set Execution Order

Function: The command opens a dialog for setting the execution position of an element to any value.

Call: Menu **CFC > Execution Order**, context menu > Execution Order

Requirements: A CFC editor is active. One element is selected.

The command only changes the chronological order in the program sequence. The position of the elements is not changed. If the command is applied to more than one element at a time, the sequence is retained within the selected elements.

See also:

- Command Order by Data Flow [▶ 192]
- Command Order By Topology [▶ 193]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.15 Command Connect Selected Pins

Symbol: 

Function: The command establishes a link between the selected pins.

Call: CFC menu, context menu

Requirements: A CFC editor is active. Exactly one output and several inputs are selected.

To select the pins, keep **[CTRL]** pressed while clicking the pins. Then execute the command.

See also:

- Command Select Connected Pins [▶ 200]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.16 Command Reset Pins

Symbol: 

Function: The command restores deleted pins of a function block.

Call: Menu **CFC > Pins**, context menu > Pins

Requirements: A CFC editor is active and a function block is selected.

The command restores all inputs and outputs of the function block, as defined in its implementation.

See also:

- Command Remove Unused Pins [▶ 194]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.17 Command Remove Unused Pins

Symbol: 

Function: The command removes all unused pins of the selected element.

Call: Menu **CFC > Pins**, context menu > Pins

Requirements: A CFC editor is active. One element is selected.

See also:

- Command Reset Pins [▶ 194]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.18 Command Add Input Pin

Symbol: 

Function: The command adds a further input to the selected function block.

Call: Menu **CFC > Pins**, context menu > Pins

Requirements: A CFC editor is active. A function block is selected.

See also:

- Command Add Output Pin [▶ 195]
- PLC documentation: Continuous Function Chart (CFC)

- PLC documentation: CFC Editor

5.12.19 Command Add Output Pin

Symbol: 

Function: The command adds a further output to the selected function block.

Call: Menu **CFC > Pins**, context menu > Pins

Requirements: A CFC editor is active. A suitable function block is selected.

See also:

- Command Add Input Pin [▶ 194]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.20 Command Route All Connections

Symbol: 

Function: The command undoes all manual changes to the connections in the program and restores the original state.

Call: Menu **CFC > Routing**, context menu > Routing

Requirements: A CFC editor is active.

TwinCAT cannot automatically route connections that are fixed with control points. You have to remove the control points before the command is executed. To do this, use the command Remove Control Point. In addition, you have to disconnect manually changed connections, which are marked with the icon  . To do this, use the command Unlock Connection.

See also:

- Command Remove Control Point [▶ 196]
- Command Unlock Connection [▶ 196]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.21 Command Show Next Collision

Function: The command indicates the next collision in the editor and marks the affected location.

Call:  button in the upper right corner of the editor

Requirements: A CFC editor is active, and there is at least one connection with collision.

This function is very useful if you work with large networks and only a subset is visible. A collision is also indicated by the symbol with a red frame in the upper right corner of the editor.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.22 Command Remove Control Point

Function: The command removes a control point.

Call: Context menu > Routing

Requirements: A CFC editor is active. You have selected a connecting line.

If you move the mouse pointer over a selected connecting line, the existing control points are displayed with yellow circle symbols. Move the cursor to the control point to be deleted and execute the command from the context menu.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor
- PLC documentation: CFC element Control point
- Command Create Control Point [▶ 196]

5.12.23 Command Create Control Point

Symbol: 

Function: The command creates a control point on a connector.

Call: Context menu > Routing

Requirements: A CFC editor is active. The cursor is over a connection.

The control point is created at the point of the connection where the cursor is placed when the command is called. The command corresponds to the **Control point** element in the **Toolbox** window.

See also:

- Command Remove Control Point [▶ 196]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor
- PLC documentation: CFC element Control point

5.12.24 Command Unlock Connection

Symbol: 

Function: This command releases a locked connection.

Call: Menu **CFC > Routing**, context menu > Routing

Requirements: A CFC editor is active. A connection or a connection mark is selected.

Changing the connections for automatic routing results in a locked connection. To perform automatic routing again, you must first release a locked connection.

You can also release this connection by clicking on the icon of a locked connection.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

- PLC documentation: CFC element Connection mark source/target

5.12.25 Command Create group

Symbol: 

Function: The command groups the selected elements.

Call: Menu **CFC > Group**, context menu > Group

Requirements: A CFC editor is active. Several elements are selected.

Grouped elements are moved together. The position of the elements is not influenced by the grouping.

See also:

- Command Ungroup [▶ 197]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.26 Command Ungroup

Symbol: 

Function: The command cancels a previously created grouping.

Call: Menu **CFC > Group**, context menu > Group

Requirements: A CFC editor is active. A grouping is selected.

See also:

- Command Create group [▶ 197]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.12.27 Command Edit page size

Function: The command opens the Edit Page Size dialog. This can be used to change the size of the page-based CFC editor.

Call: CFC menu

Requirements: A page-oriented CFC editor is active.

Dialog Edit Page Size

Width	Width of the page (minimum 24, maximum 1024). Elements outside the working area are highlighted in red.
Height	Page height (minimum 24, maximum 1024). Elements outside the working area are highlighted in red.
Border width	Margin width (minimum 6, maximum 25% or page width).
Set as default for new CFC objects	 : The current settings are set as default for new CFC objects.

See also:

- PLC documentation: Continuous Function Chart (CFC)

- PLC documentation: CFC Editor
- PLC documentation: CFC element Page

5.12.28 Command Edit Parameters

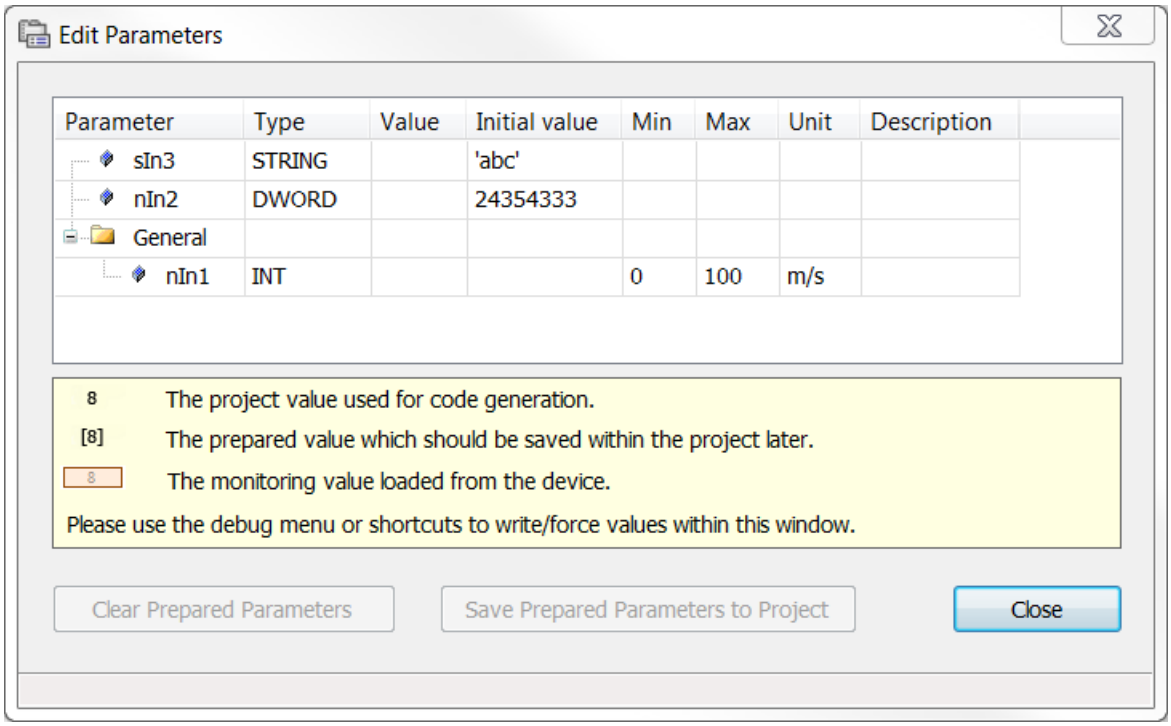
Function: The command opens the "Edit Parameters" dialog, in which you can change the constant input parameters of a function block.

Call: Menu **CFC > Edit parameters**, context menu > Edit parameters, click on function block field **Parameter**.

Requirements: A CFC editor is active. A function block is instantiated, which has VAR_INPUT CONSTANT variables in its declaration.

A function block with VAR_INPUT CONSTANT variables is indicated by TwinCAT by the word "Parameter" in the lower left corner of the function block.

Dialog Edit Parameters



Parameter	Name of the variable
Type	Data type of the variable
Value	Click in the field to enter a value.
Initial value	Initialization value
Category	Additional information about the parameters. These values are defined by attributes and cannot be changed in this dialog • parameterCategory • parameterUnit • parameterMinValue • parameterMaxValue
Unit	
Min	
Max	
Delete prepared parameters	The command is active if you have written a prepared value (command Debug > Write values)

After exiting the field and exiting the dialog with **OK**, the value changes are applied to the project.

Example of a function block with constant inputs:

```
FUNCTION_BLOCK FB_Sample
VAR_INPUT CONSTANT
    {attribute 'parameterCategory':='General'}
    {attribute 'parameterUnit':= 'm/s'}
    {attribute 'parameterMinValue':= '0'}
    {attribute 'parameterMaxValue':= '100'}
    nIn1 : INT;
    nIn2 : DWORD:=24354333;
    sIn3 : STRING:='abc';
END_VAR
```

This functionality and the declaration of variables with the keyword VAR_INPUT CONSTANT only apply to the CFC editor. In the FBD editor, TwinCAT always shows all input parameters at the function block, irrespective of whether they are declared as VAR_INPUT or VAR_INPUT CONSTANT. Also, TwinCAT does not distinguish between text editors.

See also:

- Command Save prepared parameters in the project [▶ 199]
- PLC documentation: Changing constant input parameters of function block instances

5.12.29 Command Save prepared parameters in the project

Function: The command applies the prepared parameter values in the project.

Call: CFC menu

Requirements: A CFC editor is active. Parameter values of function block instances were changed in online mode. The application is in offline mode.

If the values of constants on the controller differ from the values in the application, this is indicated by a red star to the right of the parameter field. Use the command **Apply prepared parameter values** to apply the control values to your application.

See also:

- Command Edit Parameters [▶ 198]
- PLC documentation: Changing constant input parameters of function block instances

5.12.30 Command Connection Mark

Symbol: 

Function: This command toggles the display of the connection between two elements between a connecting line and connection marks.

Call: CFC menu, context menu

Requirements: A CFC editor is active. A connection or a connection mark is selected.

If you have selected a connecting line, the command removes the line and adds a connection mark source to the output of one of the elements and a connection mark target to the input of the other. Both are assigned the same name "C-<n>" by default; n is a sequential number.

If you select a connection mark pair, the command converts these marks to a connecting line.

See also:

- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor
- PLC documentation: CFC element Connection mark source/target

5.12.31 Command Select Connected Pins

Symbol: 

Function: The command selects all pins that are connected to the currently selected line or to the currently selected connection mark in the page-oriented CFC.

Call: Context menu

Requirements: A CFC editor or a page-oriented CFC editor is active. A line or a connection mark is selected.

See also:

- Command Connect Selected Pins [▶ 193]
- PLC documentation: Continuous Function Chart (CFC)
- PLC documentation: CFC Editor

5.13 FBD/LD/IL

5.13.1 Command Insert Contact (right)

Symbol: 

Function: The command inserts a contact to the right of the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line, a contact or a box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.2 Command Insert Network

Symbol: 

Function: The command inserts another network in the FBD/LD/IL editor.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. No box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element network

5.13.3 Command Insert Network (below)

Symbol: 

Function: The command inserts another network in the FBD/LD/IL editor below the selected network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A network is selected, but no box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element network

5.13.4 Command Toggle comment state

Symbol: 

Function: The command toggles comment state of the selected network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A network is selected, but no box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.5 Command Insert Assignment

Symbol:

Function: The command inserts an assignment in the FBD or LD editor.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A network is selected, but no box is selected.

In IL, an assignment is programmed using the operators LD and ST.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element assignment

5.13.6 Command Insert Box

Symbol: 

Function: The command inserts a box that is available in the project at the end of the selected network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A network is selected, but no box is selected.

When you select the command, the Input Assistant opens. There you can select the desired box.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element function block

5.13.7 Command Insert Box with EN/ENO

Symbol: 

Function: The command inserts a box with a Boolean input "Enable" and a Boolean output "Enable Out" at the end of the selected network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A network is selected, but no box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element function block with EN/ENO

5.13.8 Command Insert Empty Box

Symbol: 

Function: The command inserts an empty box at the end of the currently selected network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A network is selected, but no box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element function block

5.13.9 Command Insert Box with EN/ENO

Symbol: 

Function: The command inserts an empty box with a Boolean input "Enable" and a Boolean output "Enable Out" at the end of the selected network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: An FBD editor, LD editor or IL editor is active. A network must be selected. No other box may be selected.

If "Enable" has the value FALSE when the box is called, the operations defined in the box are not executed. Otherwise, i.e. if "Enable" is TRUE, these operations are executed. The ENO output acts as a repeater of the EN input.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element function block with EN/ENO

5.13.10 Command Insert jump

Symbol: 

Function: The command inserts a jump element before the currently selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A connector is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element jump

5.13.11 Command Insert label

Symbol: 

Function: The command inserts a label in the currently selected network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A network is selected. No label is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element label

5.13.12 Command Insert Return

Symbol: 

Function: The command inserts a "Return" element at the selected position.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or IL editor is active. A box output is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element return

5.13.13 Command Insert Input

Symbol: 

Function: The command adds an additional input to an expandable box (ADD, OR, ADD, MUL, SEL) above the selected input.

Call: Menu **FBD/LD/IL**

Requirements: The FBD/LD editor is active. A box input is selected.

If a box is selected, the command **Append name input** is available in the context menu. The input is inserted at the bottom of the box.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.14 Command Insert box in parallel (below)

Symbol: 

Function: The command inserts an empty box in parallel to the selected box.

Call: Menu **FBD/IL/LD**, context menu

Requirements: A box is selected in the LD editor.

See also:

- PLC documentation: FBD/LD/IL Editor

5.13.15 Command Insert Coil

Symbol:

Function: The command inserts a coil in the network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A network, a coil or a connector is selected, but no box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element coil

5.13.16 Command Insert Set coil

Symbol: 

Function: The command inserts a set coil in the network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A network, a coil or a line is selected, but no box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element coil

5.13.17 Command Insert Reset coil

Symbol: 

Function: The command inserts a reset coil in the network.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A network, a coil or a line is selected, but no box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element coil

5.13.18 Command Insert Contact

Symbol: 

Function: The command inserts a contact to the left of the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line or a contact is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.19 Command Insert Contact Parallel (below)

Symbol: 

Function: The command inserts a contact with lines in parallel below the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line, a contact or a box is selected.

You can program closed parallel branches in an LD network as a Short Circuit Evaluation (SCE) or an OR construct. SCE branches are represented by double vertical lines, OR branches with single lines. See the help page for "Closed line branches“.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.20 Command Insert Contact Parallel (above)

Symbol: 

Function: The command inserts a contact with lines in parallel above the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line, a contact or a box is selected.

You can program closed parallel branches in an LD network as a Short Circuit Evaluation (SCE) or an OR construct. SCE branches are represented by double vertical lines, OR branches with single lines. See the help page for "Closed line branches“.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.21 Command Insert Negated Contact

Symbol: 

Function: The command inserts a negated contact to the left of the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line or a contact is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.22 Command Insert Negated Contact Parallel (below)

Symbol: 

Function: The command inserts a negated contact with lines in parallel below the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line, a contact or a box is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.23 Command Paste Contacts: Paste below

Function: The command pastes a previously copied contact with lines below the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active.

See also:

- PLC documentation: FBD/LD/IL

- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.24 Command Paste Contacts: Paste above

Function: The command pastes a previously copied contact with lines above the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line or a contact is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.25 Command Paste Contacts: Paste right (after)

Function: The command pastes a previously copied contact to the right of the selected element.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line or a contact is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: LD element contact

5.13.26 Command Insert IL line below

Symbol: 

Function: The command inserts an instruction line below the selected line.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The IL editor is active. A line is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.27 Command Delete IL line

Symbol: 

Function: The command deletes the selected instruction line.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The IL editor is active. A line is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.28 Command Negation

Symbol: 

Function: The command negates the following elements:

- Input/output of a box
- Jump
- Return
- Coil

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. The corresponding element is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.29 Command Edge detection

Symbol FBD: 

Symbol LD: 

Function: The command inserts an edge detection before the selected box input or box output. A distinction can be made between the following functionalities by executing the command once, twice or three times:

- The edge detection inserted when the command is executed once serves to detect a rising edge. The arrow of the symbol points to the right.
- If the command is executed again, the edge detection is reversed, so that falling edges are detected. The arrow of the symbol points to the left.
- If the command is executed one more time, the edge detection and the symbol are removed.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. A box input or output is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.30 Command Set/Reset

Symbol: 

Function: For an element with a Boolean output, the command toggles between reset, set and no label.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. An element with Boolean output is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.31 Command Set output connection

Symbol: 

Function: The command assigns the selected box output as interconnecting box output.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. One of several box outputs is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.32 Command Insert Branch

Symbol: 

Function: The command creates an open branch on the selected line.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. A box input or output is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element line branch

5.13.33 Command Insert Branch above

Symbol: 

Function: The command creates a branch above the selected open branch.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. An open branch is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element line branch

5.13.34 Command Insert Branch below

Symbol: 

Function: The command creates a branch below the selected open branch.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. An open branch is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: FBD/LD/IL element line branch

5.13.35 Command Set Branch Start Point

Symbol: 

Function: The command sets the start point of a branch on the selected line.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line is selected.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: Closed line branch

5.13.36 Command Set Branch End Point

Symbol: 

Function: The command sets the end point of a branch on the selected line.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The LD editor is active. A line is selected. A start point for the branch was set.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor
- PLC documentation: Closed line branch

5.13.37 Command Update parameters

Function: The command applies changes in the declaration of the selected element to the graphic.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD, LD or CFC editor is active. A box is selected. An extended change to the declaration was carried out.

The command checks whether a box and its declaration match in the declaration editor. The change is only applied to the box if the declaration was extended. Deletions and overwrites are not updated.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.38 Command Remove unused FB call parameters

Symbol: 

Function: The command deletes inputs and outputs of the selected function block, to which no variable and no value was assigned. However, the default inputs and outputs of the function block are always retained.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. A function block is selected. The function block has interfaces that are not assigned a value.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.39 Command Repair POU

Symbol: 

Function: The command repairs internal inconsistencies on the selected function block.

Call: Menu **FBD/LD/IL**, context menu

Requirements: The FBD or LD editor is active. The faulty function block is selected. The editor has detected internal inconsistencies in the programming block, which may be resolved automatically. TwinCAT reports the inconsistencies in the **Error list** view.

This situation is conceivable when you edit a project created with an older version of the programming system that has not yet treated the inconsistency as an error.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.40 Command View as function block diagram

NOTE

Loss of data

An error-free conversion requires syntactically correct code. Otherwise, parts of the implementation may be lost.

Function: The command converts the active Instruction List or the active Ladder Diagram to the function block diagram.

Call: Menu **FBD/LD/IL > View**

Requirements: The LD or IL editor is active.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.41 Command View as ladder logic

NOTE

Loss of data

An error-free conversion requires syntactically correct code. Otherwise, parts of the implementation may be lost.

Function: The command converts the current function block code or the active Instruction List to a Ladder Diagram.

Call: Menu **FBD/LD/IL > View**

Requirements: The FBD or IL editor is active.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.42 Command View as instruction list

If required, IL can be enabled via the TwinCAT options.

NOTE

Loss of data

An error-free conversion requires syntactically correct code. Otherwise, parts of the implementation may be lost.

Function: The command converts the active function block code or the active Ladder Diagram to an Instruction List.

Call: Menu **FBD/LD/IL > View**

Requirements: The FBD or LD editor is active.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.13.43 Command Go To

Symbol: 

Function: The command lets you jump to any network.

Call: Menu **FBD/LD/IL**

Requirements: The LD, FBD editor or IL editor is active. A network is selected.

The command opens a dialog with an input field. Enter the number of the desired network in the input field.

See also:

- PLC documentation: FBD/LD/IL
- PLC documentation: FBD/LD/IL editor

5.14 Textlist

5.14.1 Command Add Language

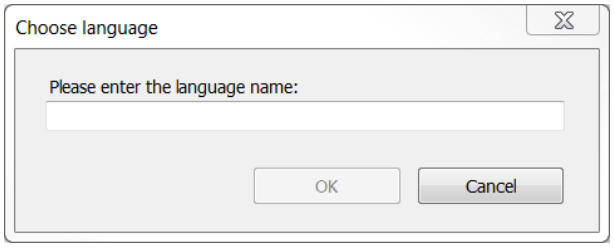
Symbol: 

Function: This command adds an additional language column in the text list.

Call: Menu **Textlist**, context menu

Requirement: A text list or a global text list is open and active.

Enter an abbreviation for the new language in the **Choose language** dialog, for example "en-US". TwinCAT inserts the abbreviation as a column heading.



See also:

- PLC documentation: Managing text in a text list

5.14.2 Command Remove Language

Symbol: 

Function: The command removes the selected language column from the text list.

Call: Menu **Textlist**, context menu

Requirement: A text list or a global text list is open and active. A field in the column of the language that you want to remove is selected.

See also:

- PLC documentation: Managing text in a text list

5.14.3 Command Insert Text

Symbol: 

Function: The command inserts a new line above the select line in the text list. An input field opens under Standard, in which you enter the source text.

Call: Menu **Textlist**

Requirement: A text list (no GlobalTextList) is open and active. A field is selected in the table.

See also:

- PLC documentation: Managing text in a text list

5.14.4 Command Import/Export Text Lists

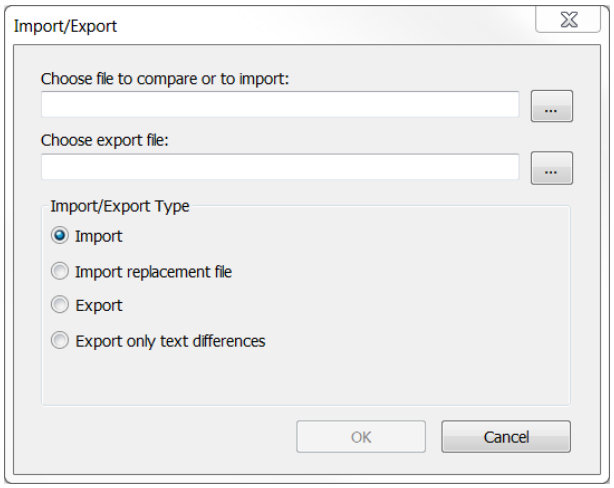
Symbol: 

Function: The command exports an active text list, imports a file or synchronizes a text list with a file. The file is in CSV format. The **Import/Export** dialog offers options for this purpose.

Call: Menu **Textlist**, context menu

Requirement: A text list or a global text list is open and active.

Import/Export dialog



Choose file to compare or to import	File, which TwinCAT reads. ... opens the **Choose textlist file** dialog, in which you can select a file.
Choose export file	File to which TwinCAT writes. ... opens the **Choose textlist file** dialog, in which you can select a file and a directory.

Import/Export Type:

Import	Requirement: A file is selected in **Choose file to compare or to import**. The file may contain text list records for the global text list of for text lists. Global text list: • TwinCAT reads the file, compares the text list records for the same source text and applies differences in the translations. TwinCAT overwrites the translations in the project, if necessary. Text lists: • TwinCAT reads the file, compares the text list records for the same IDs and applies differences in the source text and the translation to the project. TwinCAT overwrites the text list records in the project, if necessary. • If the file contains a new ID, the text list entry is imported into the text list of the project, and the text list is amended.
Import replacement file	Requirement: A replacement file is selected in **Choose file to compare or to import**. The replacement file contains replacements for the global text list. TwinCAT processes the replacement file line-by-line and implements the specified replacements in the global text list. The structure of the replacement file is described in section Managing Static Text in Global Text Lists.
Export	Requirement: The file to which TwinCAT writes is selected in "Choose export file". TwinCAT exports all texts from all text lists of the current project. All languages available in the project are inserted as columns in the export file. The file can be used to translate the language-dependent text externally.
Export only text differences	Requirement: An import file for the comparison is selected in **Choose file to compare or to import**. An export file to which TwinCAT writes is selected in **Choose export file**. TwinCAT reads the import file and compares the lines of the active text list with it. TwinCAT ignores any matching lines. If lines differ, TwinCAT writes the line to the export file and applies translations from the text list. TwinCAT applies the translations from the import file and overwrites them, if applicable.

See also:

- PLC documentation: Managing text in a text list

5.14.5 Command Remove Unused Text List Records

Symbol: 

Function: The command checks whether a text list record in the project will be used as static text. If not, TwinCAT removes it from the text list.

Call: Menu **Textlist**, context menu

Requirement: The global text list is open and active. A field is selected in the table.

See also:

- PLC documentation: Managing text in a text list

5.14.6 Command Check Visualization Text Ids

Symbol: 

Function: The command checks whether the ID of a text list record in the project is correct and reports the result.

Call: Menu **Textlist**, context menu

Requirement: The global text list is open and active. A field is selected in the table.

If TwinCAT detects that the global text list and the static texts of the visualizations do not match, this may be because the global text list is or was read-only. A requirement is that you have set up user management in the project.

See also:

- PLC documentation: Managing text in a text list

5.14.7 Command Update Visualization Text Ids

Symbol: 

Function: The command updates all inconsistent IDs in a static text list.

Call: Menu **Textlist**, context menu

Requirement: The global text list is open and active. A field is selected in the table. The object is read-only.

If TwinCAT detects that the global text list and the static texts of the visualizations do not match, this may be because the global text list is or was read-only. A requirement is that you have set up user management in the project.

See also:

- PLC documentation: Managing text in a text list

5.14.8 Command Export All

Symbol: 

Function: The command exports all text lists of the project.

Call: Menu **Textlist**, context menu

Requirement:

- A text list or a global text list is open and active.
- The visualization does not encode the text characters in Unicode.

For each text list TwinCAT creates a file as plain text in .txt format. The name of text list becomes the name of the file. The file is saved in the directory of the TwinCAT project.

A controller can read and use this format. You can copy the file to a controller, for example, and configure it via a setting in the Visualization Manager, such that the text lists are not transferred again when the PLC project is loaded.

See also:

- PLC documentation: Managing text in a text list

5.14.9 Command Export All Unicode

Symbol: 

Function: The command exports all text lists of the project.

Call: Menu **Textlist**, context menu

Requirement:

- A text list or a global text list is open and active.
- The visualization encodes the text characters in Unicode.
  - The option **Use Unicode strings** is enabled in the Visualization Manager.

For each text list TwinCAT creates a file as plain text in .txt format. The name of text list becomes the name of the file. The file is saved in the directory of the TwinCAT project.

A controller can read and use this format. You can copy the file to a controller, for example, and configure it via a setting in the Visualization Manager, such that the text lists are not transferred again when the PLC project is loaded.

See also:

- PLC documentation: Managing text in a text list

5.14.10 Command Add text list support

Symbol: 

Function: The command adds text list support to the selected DUT object of type **Enumeration**.

Call: Context menu of a standard DUT object of type **Enumeration** ()

Text list support enables localization of the enumeration component identifiers and a representation of the symbolic component value in a text output in the visualization.

See also:

- PLC documentation: Object DUT
- Command Remove text list support [▶ 217]

5.14.11 Command Remove text list support

Symbol: 

Function: The command removes text list support from the selected enumeration object.

Call: Context menu of an enumeration object with text list support ().

Text list support enables localization of the enumeration component identifiers and a representation of the symbolic component value in a text output in the visualization.

5.15 Recipes

5.15.1 Command Add a new recipe

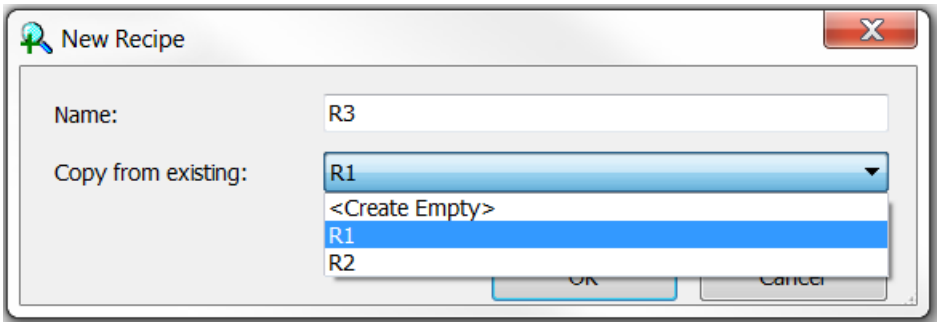
Symbol: 

Function: The command opens a dialog for inserting a new recipe (a new column) in the recipe definition.

Call: Menu **Recipes**, context menu

Requirement: A recipe definition is open in the editor.

When you run the command, a dialog opens where you can set the name of the new recipe. The dialog also offers an option for copying existing recipes into the new recipe.



See also:

- PLC documentation: Changing values with recipes

5.15.2 Command Remove recipe

Symbol: 

Function: This command deletes a recipe from the currently open recipe definition.

Call: Menu **Recipes**, context menu

Requirement: A field is selected in the recipe column of a recipe definition.

See also:

- PLC documentation: Changing values with recipes

5.15.3 Command Load Recipe

Symbol: 

Function: The command loads a recipe from a file.

Call: Menu **Recipes**, context menu

Requirement: A field is selected in the recipe column of a recipe definition.

When you run the command, the standard dialog for selecting a file opens. The filter is automatically set to the file extension *.txtrecipe. After loading, the values of the selected recipe in the recipe definition are overwritten, and the display is updated.

If you want to overwrite only individual recipe variables with new values, remove the values for the other variables before loading the recipe into the recipe file. Entries without value specification are not read, which means that these variables are unaffected by the update on the controller and in the project. The following is an example of the entries in a recipe file. When it is loaded, only MAIN.nVar is written with a new value (6):

```
MAIN.nVar1:=
MAIN.nVar2:=6
MAIN.nVar3:=
MAIN.sVar4:=
MAIN.wsVar5:=
```

See also:

- PLC documentation: Changing values with recipes

5.15.4 Command Save recipe

Symbol: 

Function: The command saves the values of recipe variables in a file.

Call: Menu **Recipes**, context menu

Requirement: A recipe value is selected in the recipe definition.

When the command is executed, TwinCAT saves the values of the selected recipe in a file with the extension *.txtrecipe, whose name must be defined. The standard dialog for saving a file opens. The format results from the settings of the Recipe Manager in the **Storage** tab.

Overwriting the implicit recipe file

The implicitly used recipe files, which are needed as a clipboard for reading and writing recipes, must not be overwritten. This means that the file name must be different to <Recipe name>.<Recipe definition name>.txtrecipe

See also:

- PLC documentation: Changing values with recipes

5.15.5 Command Read Recipe

Symbol: 

Function: The command reads the variable values of a recipe from the controller.

Call: Menu **Recipes**, context menu

Requirement: The PLC project is in online mode, and a recipe value is selected in the recipe definition.

When the command is executed, TwinCAT overwrites the values of the selected recipe with the values read from the controller. In the process, the values are implicitly stored (in a file on the controller) and simultaneously displayed in the table of the recipe definition.

See also:

- PLC documentation: Changing values with recipes

5.15.6 Command Write Recipe

Symbol: 

Function: The command writes the values of a recipe to the variables in the controller.

Call: Menu **Recipes**, context menu

Requirement: The PLC project is in online mode, and a recipe value is selected in the recipe definition.

When the command is executed, TwinCAT overwrites the values in the controller with the values of the selected recipe.

See also:

- PLC documentation: Changing values with recipes

5.15.7 Command Load and Write Recipe

Symbol: 

Function: This command loads a recipe from a file and writes the values to the variables in the controller.

Call: Menu **Recipes**, context menu

Requirement: The PLC project is in online mode, and a recipe value is selected in the recipe definition.

After running the command, you are asked whether the values from the file should also be written to the recipe in the project, or only to the PLC. An update of the values in the recipe may necessitate an Online Change with the next login.

When you execute the command, TwinCAT overwrites the values of the selected recipe in the recipe definition. In addition, the values of variables in the controller are overwritten with these recipe values.

If you want to overwrite only individual recipe variables with new values, remove the values for the other variables before loading the recipe into the recipe file. Entries without value specification are not read, which means that these variables are unaffected by the update on the controller and in the project. The following is an example of the entries in a recipe file. When it is loaded, only MAIN.nVar1 is written with a new value (6).

```
MAIN.nVar1:=
MAIN.nVar2:=6
MAIN.nVar3:=
MAIN.sVar4:=
MAIN.wstVar5:=
```

See also:

- PLC documentation: Changing values with recipes

5.15.8 Command Read and Save Recipe

Symbol: 

Function: The command reads the variables values of a recipe from the controller and saves them in a file.

Call: Menu **Recipes**, context menu

Requirement: The PLC project is in online mode, and a recipe value is selected in the recipe definition.

After executing the command you are asked whether to read the variables values into the recipe in the project, or only save them. An update of the values in the recipe may necessitate an Online Change with the next login.

The values are saved with the default name for the recipe files, in accordance with the settings of the Recipe Manager (**Storage** tab).

See also:

- PLC documentation: Changing values with recipes

5.15.9 Command Insert variable

Symbol: 

Function: The command adds a variable to the currently open recipe definition before the selected position.

Call: Menu **Recipes**, context menu

Requirement: The recipe definition is open in the editor, and simple view is selected.

TwinCAT adds the default text "NewVariable" in the **Variable** column. You must replace this name with the corresponding valid variable name. To this end, open the Input Assistant via the ... button, or enter the variable name directly in the table field.

See also:

- PLC documentation: Changing values with recipes

5.15.10 Command Remove variables

Symbol 

Function: The command removes the selected variable(s) from a recipe definition.

Call: [Del] key, context menu

Requirement: You have selected a variable.

See also:

- PLC documentation: Changing values with recipes

5.15.11 Command Update structured variables

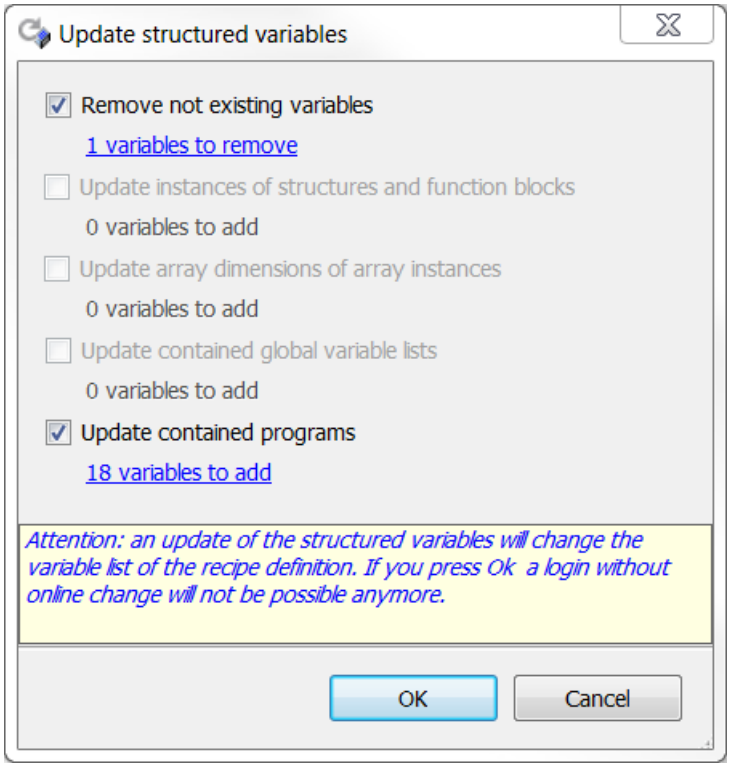
Symbol: 

Function: The command opens the dialog **Update structured variables**.

Call: Menu **Recipes**

In the dialog you can update the recipe definitions if the declaration of a structured variable or a function block has changed. If, for example, the dimension of an array has been changed, you can automatically remove or add the entries in the recipe definition.

Dialog Update structured variables



Remove non existing variables	: Variables that no longer exist in the project due a change in a structured element are removed from the recipe definition.
Update instances of structures and function blocks	: If the declaration of a structure or function block represented by an instance in the recipe definition is extended, the corresponding variables are added to the recipe definition.
Update array dimensions of array instances	: If the dimension of an array is extended, which is represented with an instance in the recipe definition, the corresponding variables are added to the recipe definition.
Update contained global variable lists	: If the declaration of a global variable list represented by an instance in the recipe definition is extended, the corresponding variables are added to the recipe definition.
Update contained programs	: If the declaration of a program is extended, which was instantiated in the recipe definition, the corresponding variables are added to the recipe definition.

See also:

- PLC documentation: Changing values with recipes

5.15.12 Command Download recipes from the device

Symbol: 

Function: The command initiates the synchronization of the recipes of the currently open recipe definition in the project with the recipes that are present on the device in the form of recipe files.

Call: Menu **Recipes**

Requirement: The PLC project is in online mode, and you have opened a recipe definition in the editor. In detail, the synchronization has the following effect:

- The current values for the recipe variables present in the project are overwritten with the values from the recipes on the controller. This may trigger an Online Change at the next login.
- If recipe variables are defined in the recipe files on the controller, which are missing in the recipe definition in the project, these variables are ignored on loading. For each recipe file, a message with the affected variables appears.
- If recipe variables are missing in the recipe files on the controller, which are included in the recipe definition in the project, for each recipe file a message appears showing the affected variables.
- If additional recipes were created on the controller for these variables, these are added to the recipe definition in the project.

5.16 Library

Command	Further information the PLC documentation
Library creation	
Command Save as library...	Command Save as library [▶ 245]
Command Save as library and install...	Command Save as library and install [▶ 247]
Library installation	
Command Library Repository	Library Repository
Library management	
Dialog Library Manager	Library Manager
Other commands and dialogs	
Command Add library	Command Save as library and install [▶ 247]
Command Try reload library	Command Try reload library
Command Delete library	Command Delete library
Command Details	Command Details
Command Dependencies	Command Dependencies
Command Properties	Command Properties
Command Placeholder	Placeholder
Command Set to Effective Version	Command Set to Effective Version
Command Set to Always Newest Version	Command Set to Always Newest Version

See also:

- PLC documentation: Using libraries > Other commands and dialogs

5.17 Visualization

The visualization commands are provided by the **Visual Editor** plug-in for the visualization commands menu category, which can be found in the dialog **Tools > Customize**. They enable you to edit a visualization object in the visualization editor.

Most commands are included in the **Visualization** menu as standard, and are therefore also available in the context menu of the visualization editor. If necessary, open the dialog **Tools > Customize**, in order to view or modify the menu configuration for the category **Visualization**.

5.17.1 Command Interface Editor

Symbol: 

Function: This command opens the Interface Editor for defining frame parameters in the visualization, which are intended to be referenced in the **Frame** element of another visualization. It is displayed in a tab view in the upper section of the Visualization Editor.

Call: Menu **Visualization**

5.17.2 Command Hotkey Configuration

Symbol: 

Function: This command opens the keyboard configuration editor for the current visualization. It is displayed in a tab view in the upper section of the Visualization Editor.

Call: Menu **Visualization**

5.17.3 Command Element List

Symbol: 

Function: This command opens the element list editor for the current visualization. It is displayed in a tab view in the upper section of the Visualization Editor.

Call: Menu **Visualization**

5.17.4 Command Align Left

Symbol: 

Function: This command aligns all selected visualization elements at the left edge of its leftmost element.

Call: Menu **Visualization**, context menu

5.17.5 Command Align Top

Symbol: 

Function: This command aligns all selected visualization elements at the top edge of its topmost element.

Call: Menu **Visualization**, context menu

5.17.6 Command Align Right

Symbol: 

Function: This command aligns all selected visualization elements at the right edge of its rightmost element.

Call: Menu **Visualization**, context menu

5.17.7 Command Align Bottom

Symbol: 

Function: This command aligns all selected visualization elements at the bottom edge of its bottommost element.

Call: Menu **Visualization**, context menu

5.17.8 Command Align Vertical Center

Symbol: 

Function: This command causes all selected visualization elements to be aligned with their common vertical center.

Call: Menu **Visualization**, context menu

5.17.9 Command Align Horizontal Center

Symbol:

Function: This command causes all selected visualization elements to be aligned with their common horizontal center.

Call: Menu **Visualization**, context menu

5.17.10 Command Make horizontal spacing equal

Symbol: 

The command becomes active if three or more elements are selected.

1. Select all the elements to be positioned with the same horizontal spacing.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Make horizontal spacing equal**.

⇨ The elements are positioned such that the leftmost and rightmost elements retain their positions and the elements in between are aligned with equal horizontal spacing.

5.17.11 Command Increase horizontal spacing

Symbol: 

The command becomes active if two or more elements are selected.

1. Select all the elements to be positioned with increased horizontal spacing.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Increase horizontal spacing**.

⇨ The elements are positioned such that the blue element retains its position and the other elements are aligned horizontally with more spacing between the elements. The spacing increases by 1 pixel.

5.17.12 Command Decrease horizontal spacing

Symbol: 

The command becomes active if two or more elements are selected.

1. Select all the elements to be positioned with reduced horizontal spacing.
2. Run the command **Decrease horizontal spacing**.

⇨ The elements are positioned such that the blue element retains its position and the other elements are aligned horizontally with reduced spacing between the elements. The spacing decreases by 1 pixel.

5.17.13 Command Remove horizontal spacing

Symbol: 

The command becomes active if two or more elements are selected.

1. Select all the elements to be positioned without horizontal spacing.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.

2. Run the command **Remove horizontal spacing**.

⇨ The elements are positioned such that the blue element retains its position and the other elements are aligned horizontally without space between each other.

5.17.14 Command Make vertical spacing equal

Symbol: 

The command becomes active if two or more elements are selected.

1. Select all the elements to be positioned with the same vertical spacing.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Make vertical spacing equal**.

⇨ The elements are positioned such that the topmost and bottommost elements retain their positions and the elements in between are aligned with equal vertical spacing.

5.17.15 Command Increase vertical spacing

Symbol: 

The command becomes active if two or more elements are selected.

1. Select all the elements to be positioned with increased vertical spacing.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Increase vertical spacing**.

⇨ The elements are positioned such that the blue element retains its position and the other elements are aligned vertically with more spacing between the elements. The spacing increases by 1 pixel.

5.17.16 Command Decrease vertical spacing

Symbol: 

The command becomes active if two or more elements are selected.

1. Select all the elements to be positioned with reduced vertical spacing.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Decrease vertical spacing**.

⇨ The elements are positioned such that the blue element retains its position and the other elements are aligned vertically with reduced spacing between the elements. The spacing decreases by 1 pixel.

5.17.17 Command Remove vertical spacing

Symbol: 

The command becomes active if two or more elements are selected.

1. Select all the elements to be positioned without vertical spacing.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Remove vertical spacing**.

⇨ The elements are positioned such that the blue element retains its position and the other elements are aligned vertically without spacing between each other.

5.17.18 Command Make same width

Symbol: 

The command becomes active if more than one element is selected, except if a line or polygon element is selected.

1. Select all the elements that should have the same width.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Make same width**.

⇨ All elements are assigned the width of the element marked in blue.

5.17.19 Command Make same height

Symbol: 

The command becomes active if more than one element is selected, except if a line or polygon element is selected.

1. Select all the elements that should have the same height.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Make same height**.

⇨ All elements are assigned the height of the element marked in blue.

5.17.20 Command Make same size

Symbol: 

The command becomes active if more than one element is selected, except if a line or polygon element is selected.

1. Select all the elements that should have the same size.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Make same size**.

⇨ All elements are assigned the size of the element marked in blue.

5.17.21 Command Size to Grid

Symbol: 

The command becomes active if more than one element is selected, except if a line or polygon element is selected.

1. Select all the elements whose position and size should be aligned to the grid.
   ⇨ The first element is highlighted in blue, the other elements are shown in gray.
2. Run the command **Size to Grid**.

⇨ All elements are aligned to the grid, according to their size and position.

5.17.22 Command Bring One to Front

Symbol: 

Function: This command positions the selected element one level higher, i.e. further in the foreground of the visualization. Elements at lower levels are concealed by those at higher levels.

Call: Menu **Visualization**, context menu

5.17.23 Command Bring to front

Symbol: 

Function: This command positions the selected element in the foreground of the visualization, i.e. at the highest level. Elements at lower levels are concealed by those at higher levels.

Call: Menu **Visualization**, context menu

5.17.24 Command Send One to Back

Symbol: 

Function: This command positions the selected element one level lower, i.e. further in the background of the visualization. Elements at lower levels are concealed by those at higher levels.

Call: Menu **Visualization**, context menu

5.17.25 Command Send to Back

Symbol: 

Function: This command positions the selected element in the background of the visualization, i.e. at the lowest level. Elements at lower levels are concealed by those at higher levels.

Call: Menu **Visualization**, context menu

5.17.26 Command Group

Symbol: 

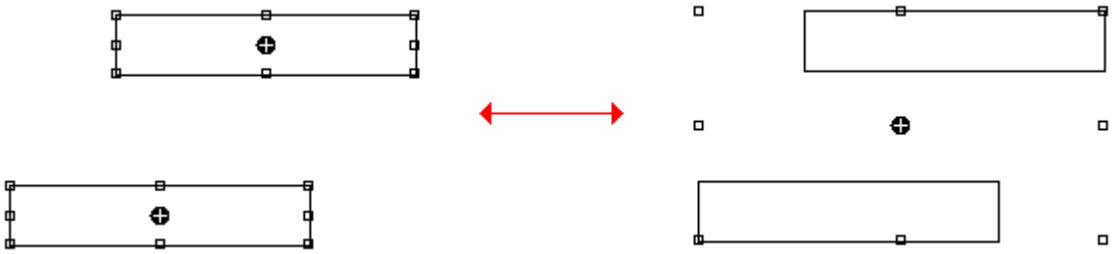
Function: This command groups the currently selected visualization elements and displays the group as a single object.

Call: Menu **Visualization**, context menu

Requirement: Several visualization elements are selected. For multiple selections hold down the **[<Shift>]** key while you click the items that you want. Alternatively, you can click in the editor window outside an element and draw a rectangle around the desired elements while pressing the mouse button.

Use the command **Ungroup** to break up the group.

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



- Command Ungroup [▶ 229]

5.17.27 Command Ungroup

Symbol: 

Function: This command breaks up a group of visualization elements. The individual elements are shown individually selected again.

Call: Menu **Visualization**, context menu

See also:

- Group [▶ 228]

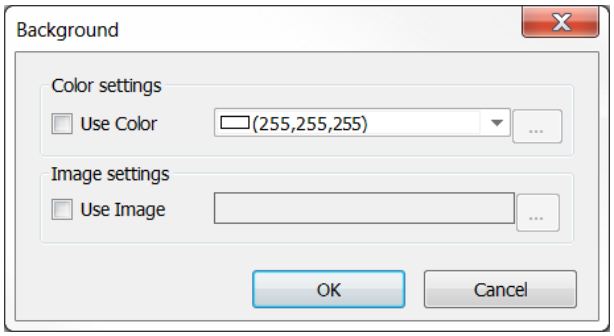
5.17.28 Command Background

Symbol: 

Function: This command opens the dialog **Frame Configuration**, in which a color or an image for the visualization background can be selected.

Call: Menu **Visualization**, context menu

Dialog Background



Enable the desired option(s):

- Bitmap: To define a background image, enter the path of an image file, which is available in an image pool in the project. Enter the name of the image pool and the image file ID, separated by a dot ".": <image pool>.<ID> (e.g. "Images_1.drive_icon>", "Images_1.43").
- Color: To define the background color of the visualization, select the desired color from the color list.

5.17.29 Command Select All

Symbol:  / 

Function: This command selects all the elements of the visualization that is currently open in the editor.

Call: Menu **Visualization**, context menu

See also:

- Deselect All [▶ 230]

5.17.30 Command Deselect All

Symbol: 

Function: This command clears the current selection of visualization elements.

Call: Menu **Visualization**

See also:

- Command Select All [▶ 229]

5.17.31 Command Multiply visu element

Symbol: 

Function: The command opens the dialog **Multiply visu element**, which you can use to configure the multiplying procedure.

Call: Menu **Visualization**, context menu

Requirement: The visualization is active, and a template element is selected.

Dialog Multiply visu element

TwinCAT adds more elements, which resemble the template element. In this dialog you can configure the number and arrangement of the indices, as well as their replacement.

Tab Basic settings

Total number of elements:

TwinCAT inserts the new elements as a table. The number of lines is set in **horizontal**, the number of columns in **vertical**. The product of the two determines the actual total number of elements to be inserted.	
Horizontal	Number of elements per line Preset: Corresponds to the number of components in $FIRSTDIM$
Vertical	Number of elements per column Preset: Corresponds to the number of components in $SECONDDIM$

Offset between the elements:

TwinCAT arranges the elements in the visualization as a table. If you specify an offset, a spacing is inserted between the elements. • 0 : The element frames overlap by one pixel • 1 : The elements are in contact • <n> : Between the elements there is a visible spacing of n-1 pixels	
Horizontal	Line spacing of elements in pixels
Vertical	Column spacing of elements in pixels

Advanced Settings tab

First dimension:

Start index	Start index for $FIRSTDIM$ Preset: 1 means the specific index for $FIRSTDIM$ starts with 1. Example array[1, <$SECONDDIM$>]
Increment	Number by which the index is incremented Preset: 1

Second dimension:

Start index	Starting index for $SECONDDIM$ Preset: 1 means the specific index for $SECONDDIM$ starts with 1. Example array [<$FIRSTDIM$>, 1] are relevant
Increment	Number by which the index is incremented Preset: 1

5.17.32 Command Activate keyboard usage

Symbol: 

Function: This command is available in the menu bar for an integrated visualization (diagnostic visualization). It enables or disables keyboard operation in the online mode of a visualization.

Call: Menu **Visualization**

If keyboard operation is enabled, elements can be entered and selected via certain shortcuts. In this case, other keyboard commands are not executed as long as the visualization editor is active and online.

5.18 Miscellaneous

5.18.1 Command Implement interfaces

Function: The command updates the implemented interfaces for a function block by adding the interface elements that the function block does not currently contain.

Call: Context menu when the function block is selected in the PLC project tree.

Requirement: The function block implements an interface that you have changed. For example, you have added another method to the interface.

Applications

When this command is executed, the automatically created methods or properties are assigned a pragma attribute, which provokes compilation errors or warnings. This provides supports in the sense that automatically created elements do not remain empty unintentionally. Whether an error or warning attribute is used depends on the application.

Case 1:

Situation: The function block for which the command **Implement interfaces** is executed is **not** derived from another function block.

Consequence: When the command is executed, the interface elements are created in the function block without implementation ("stubs") and assigned a warning attribute (in the first line of the method/property declaration). The warnings generated during compilation alert you to the fact that these elements have been generated automatically and that you need to add the required implementation code.

```
{warning 'add method/property implementation'}
```

Procedure: Add the desired implementation code to the corresponding interface element (method or property). Then remove the warning attribute from the method or property declaration.

Case 2:

Situation: The function block for which the command Implement interfaces is executed is derived from another function block. The element (method or property) created when the command is executed in the derived function block is **not** provided by inheritance from the base function block (that is, the element does not exist under the base function block or a higher parent class).

Consequence/procedure: see case 1.

Case 3:

Situation: The function block for which the command **Implement interfaces** is executed is derived from another function block. The element (method or property) created when the command is executed in the derived function block is already provided by inheritance from the base function block (that is, the element exists under the base function block or a higher parent class).

Consequence: When the command is executed, the interface element is created in the derived function block without implementation ("stub") and assigned an error attribute (in the first line of the method/property declaration). The error generated during compilation alerts you to the fact that this interface element was generated automatically and that this method or property overwrites the corresponding element of the basic function block.

```
{error 'add method/property implementation or delete method/property to use base implementation'}
```

Procedure: If you want to overwrite or enhance the method or property of the basic function block, add the required implementation code to the element below the derived function block. Then remove the error attribute from the method or property declaration. However, if you do **not** want to overwrite the method or property of the basic function block, delete the method or property below the derived function block. In this case the method or property implementation of the basic function block is used.

5.19 Context menu TwinCAT project

5.19.1 Command Save <TwinCAT project name> as Archive...

Symbol: 

Function: The command opens the standard dialog for saving a file as an archive. The project can be stored under the desired path as a ***.tszip** archive.

Call: **File** menu, context menu

Requirement: The TwinCAT project is selected in the **Solution Explorer**.

Content of *.tszip	The *.tszip archive folder contains the TwinCAT project to be archived.
Command for opening	A tszip archive can be reopened with the following command: Command Project/Solution (Open Project/Solution) [▶ 51]
Note on PLC projects	If the TwinCAT project contains one or more PLC projects, the files and folders stored in the archive folder for those PLC projects will depend on the PLC project settings of the respective project. Settings tab [▶ 112]

5.19.2 Command Send by E-Mail...

Symbol: 

Function: The command starts the email program that is set in the system and opens a new email with the archive file of the selected project as an attachment.

Call: File menu, context menu

5.19.3 Command Backup <TwinCAT project names> automatically to the target system

Function: This command can be used to enable or disable automatic storage of the TwinCAT project as .tszip on the target system during activation. This is necessary if you want to load and open the project from the target system at a later time using the command Command Open Project from Target [▶ 59].

Call: Context menu of the TwinCAT project

Requirement: The TwinCAT project is selected in the Solution Explorer.

5.19.4 Command Compare <TwinCAT project name> with the target system...

Function: This command enables the selected TwinCAT project to be compared with the project status on the target system using the TwinCAT Project Compare tool.

Call: Context menu of the TwinCAT project

Requirement: The TwinCAT project is selected in the Solution Explorer and a target system is selected to which a TwinCAT project has already been downloaded.

See also:

- Command Update project with target system... [▶ 233]

5.19.5 Command Update project with target system…

Function: This command enables the selected TwinCAT project to be updated to the project status of the connected target system. A pop-up dialog with a list of the changed files is opened for this purpose, which must be confirmed in order for the action to be successful. A detailed comparison is not provided here.

Call: Context menu of the TwinCAT project

Requirement: The TwinCAT project is selected in the Solution Explorer.

See also:

- Command Compare <TwinCAT project name> with the target system... [▶ 233]

5.19.6 Command Load project with TwinCAT 2.xx Version…

Function: This command opens the standard browser dialog, through which a TwinCAT 2 System Manager file can be selected and imported. In this way you can convert an existing TwinCAT 2 project, including the System Manager settings and the PLC project, to TwinCAT 3.

Call: Context menu of the TwinCAT project

Requirement: The TwinCAT project has been newly created without changes and selected in the Solution Explorer.

See also:

- Open a TwinCAT 2 PLC project

5.19.7 Command Show Hidden Configurations

Function: This command opens a detail menu in which the hidden configurations are listed and can be displayed again.

Call: Context menu of the TwinCAT project

Requirement: The TwinCAT project is selected in the Solution Explorer.

5.19.8 Command Remove From Solution

Symbol: 

Function: This command enables the TwinCAT project to be deleted from the Solution.

Call: Context menu of the TwinCAT project

Requirement: The TwinCAT project is selected in the Solution Explorer.

5.19.9 Command Rename

Symbol: 

Function: This command enables the TwinCAT project to be renamed in the **Solution Explorer**.

Call: Context menu of the TwinCAT project

Requirement: The TwinCAT project is selected in the Solution Explorer.

5.19.10 Command Build TwinCAT project

Symbol: 

Function: This command starts the compilation process or the code generation for the currently active TwinCAT project.

Call: **Build** menu if a TwinCAT project is currently selected, or context menu of the TwinCAT project

Requirement: The TwinCAT project is selected.

All of the projects (PLC, C++, etc.) contained in the TwinCAT project are compiled one after the other. The steps performed for a PLC project are described in section Command Build PLC project [▶ 242].

See also:

- Command Rebuild a TwinCAT project [▶ 234]

5.19.11 Command Rebuild a TwinCAT project

Function: The command starts the compilation process or the code generation for the currently active TwinCAT project, even if it was last compiled without error.

Call: **Build** menu if a TwinCAT project is currently selected, or context menu of the TwinCAT project

Requirement: The TwinCAT project is selected.

When rebuilding the project, the TwinCAT project will first be cleaned (see also: Command Clean TwinCAT project [▶ 235] and subsequently built (see also: Command Build TwinCAT project [▶ 234]).

5.19.12 Command Clean TwinCAT project

Function: This command deletes the local compilation information for the currently active PLC project and updates the language model of all objects.

Call: **Build** menu if a TwinCAT project is currently selected, or context menu of the TwinCAT project

Requirement: The TwinCAT project is selected.

All of the projects (PLC, C++, etc.) contained in the TwinCAT project are cleaned one after the other. The steps executed for a PLC project are described in the section Command Clean PLC project [▶ 243].

See also:

- Command Rebuild a TwinCAT project [▶ 234]

5.19.13 Command Unload Project

Function: This command unloads the TwinCAT project so that all files of the TwinCAT project are released.

Call: Project menu or context menu of the TwinCAT project

Requirement: The TwinCAT project is selected in the Solution Explorer.

5.19.14 Import via AML DataExchange...

Function: The command opens the standard browser dialog, via which you can search for and import a file in AutomationML format.

Call: The command can be called from the context menu of the TwinCAT project under **Import AutomationML** or via the **TwinCAT** item in the menu bar under **AutomationML** and **Import AutomationML**.

Requirement: The TwinCAT project is selected in the Solution Explorer.

See also

- Command: Open AML DataExchange Log (local)

5.19.15 Export AutomationML...

Function: The command opens the standard browser dialog for saving a file in AutomationML format for exporting the topology under the I/O node.

Call: The command can be called from the context menu of the TwinCAT project or via the **TwinCAT** item in the menu bar under **AutomationML**.

Requirement: The TwinCAT project is selected in the Solution Explorer.

5.20 PLC project context menu

5.20.1 Command Activate boot project

Symbol: 

Function: This command generates or updates the boot project of the target system.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

5.20.2 Command Autostart boot project

Function: This command changes the option **Autostart boot project** in the PLC project. If the option was previously deactivated, it is activated after the command has been executed. If the option was previously activated, it is deactivated after the command has been executed.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

5.20.3 Command Change ADS port

Function: You can use a dialog that opens to change the ADS port number of the PLC runtime system. The default setting of the ADS port for the first PLC runtime system is 851, for the second 852, etc.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

If the PLC project has multiple instances (see Command Add New Item (instance) [▶ 238]), the dialog to change the ADS port appears in succession as many times as the number of project instances. Use the first dialog to set the ADS port of the first project instance, the second dialog to set the ADS port of the second project instance, and so on.

5.20.4 Command Install Project Library

Function: The libraries that are contained in the _Libraries folder at the file level of the PLC project are installed in the local library repository. Subsequently, all libraries contained in the repository are known and can be used in projects. If a library was already known in the local repository before the execution of the command, the library will be re-installed. In the normal case it is therefore sufficient to execute the command Install project libraries (unknown versions) [▶ 237].

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

Background information:

The _Libraries folder contains the libraries that are referenced directly or indirectly in the PLC project. Direct referencing means a library that is integrated at the highest level within a library manager. Indirect referencing means a library that is referenced within a different library.

The purpose of the folder is, for example, to archive the libraries used in a project within the scope of a project archive (you can configure whether this should be the case in the project settings, see Settings tab [▶ 112]). This allows you to distribute the project archive to a colleague, for example, and if the project uses libraries that do not exist in the colleague's local library repository, the missing libraries can be installed by using the command Install project libraries [▶ 236] or Install project libraries (unknown versions) [▶ 237].

Recently used libraries are added to the _Libraries folder at the following times:

- when creating a file/email archive [▶ 50], if the file/e-mail archive is configured to contain libraries (see Settings tab [▶ 112]).
- when activating the configuration [▶ 123] or activating the boot project [▶ 236], if the target archive is configured to contain libraries (see Settings tab [▶ 112]).
- when executing the command Update project library folder [▶ 238].

Libraries that are no longer in use are removed from the _Libraries folder at the following times:

- when executing the command Update project library folder [▶ 238].

See also:

- Command Install Project Library (unknown versions) [▶ 237]
- Command Update project library folder [▶ 238]
- PLC documentation: Using libraries

5.20.5 Command Install Project Library (unknown versions)

Function: The libraries that are contained in the _Libraries folder at the file level of the PLC project and have not been part of the local library repository until now are installed in the local library repository. Subsequently, all libraries contained in the repository are known and can be used in projects. If a library was already known in the local repository before the execution of the command, it will not be re-installed.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

Background information:

The _Libraries folder contains the libraries that are referenced directly or indirectly in the PLC project. Direct referencing means a library that is integrated at the highest level within a library manager. Indirect referencing means a library that is referenced within a different library.

The purpose of the folder is, for example, to archive the libraries used in a project within the scope of a project archive (you can configure whether this should be the case in the project settings, see Settings tab [▶ 112]). This allows you to distribute the project archive to a colleague, for example, and if the project uses libraries that do not exist in the colleague's local library repository, the missing libraries can be installed by using the command Install project libraries [▶ 236] or Install project libraries (unknown versions) [▶ 237].

Recently used libraries are added to the _Libraries folder at the following times:

- when creating a file/email archive [▶ 50], if the file/e-mail archive is configured to contain libraries (see Settings tab [▶ 112]).
- when activating the configuration [▶ 123] or activating the boot project [▶ 236], if the target archive is configured to contain libraries (see Settings tab [▶ 112]).
- when executing the command Update project library folder [▶ 238].

Libraries that are no longer in use are removed from the _Libraries folder at the following times:

- when executing the command Update project library folder [▶ 238].

See also:

- Command Install Project Library [▶ 236]

- Command Update project library folder [▶ 238]
- PLC documentation: Using libraries

5.20.6 Command Update project library folder

Function: The _Libraries folder at the file level of the PLC project are updated. Subsequently, the folder contains all the libraries that are used in the PLC project. Libraries that were previously contained in the _Libraries folder, but are now no longer in use are removed from the folder.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

Background information:

The _Libraries folder contains the libraries that are referenced directly or indirectly in the PLC project. Direct referencing means a library that is integrated at the highest level within a library manager. Indirect referencing means a library that is referenced within a different library.

The purpose of the folder is, for example, to archive the libraries used in a project within the scope of a project archive (you can configure whether this should be the case in the project settings, see Settings tab [▶ 112]). This allows you to distribute the project archive to a colleague, for example, and if the project uses libraries that do not exist in the colleague's local library repository, the missing libraries can be installed by using the command Install project libraries [▶ 236] or Install project libraries (unknown versions) [▶ 237].

Recently used libraries are added to the _Libraries folder at the following times:

- when creating a file/email archive [▶ 50], if the file/e-mail archive is configured to contain libraries (see Settings tab [▶ 112]).
- when activating the configuration [▶ 123] or activating the boot project [▶ 236], if the target archive is configured to contain libraries (see Settings tab [▶ 112]).
- when executing the command Update project library folder [▶ 238].

Libraries that are no longer in use are removed from the _Libraries folder at the following times:

- when executing the command Update project library folder [▶ 238].

See also:

- Command Install Project Library [▶ 236]
- Command Install Project Library (unknown versions) [▶ 237]
- PLC documentation: Using libraries

5.20.7 Command Change project

Function: This command opens the standard browser dialog, via which you can search for and select a PLC project file. The currently selected PLC project will subsequently be replaced by the newly selected PLC project.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

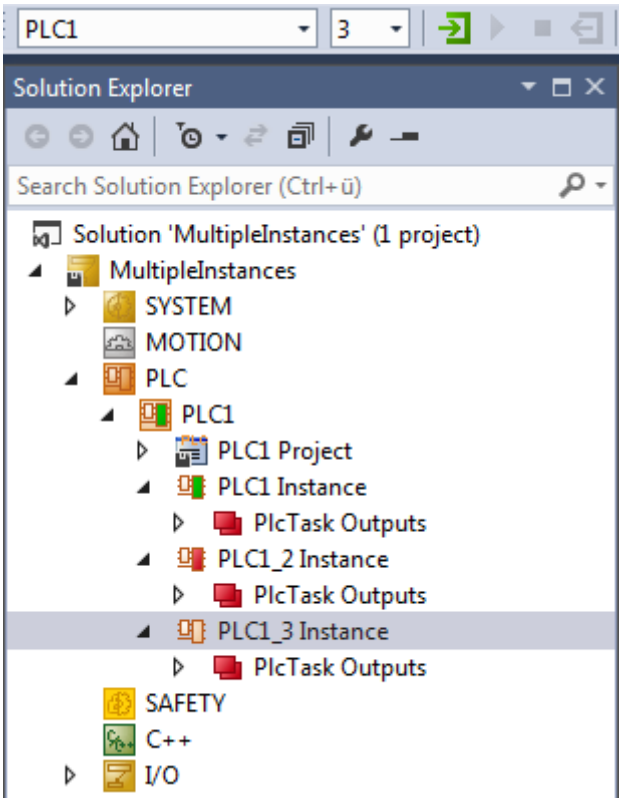
5.20.8 Command Add New Item (instance)

Function: This command can be used to add another instance (process image) to the selected PLC project ("Multiple PLC instances").

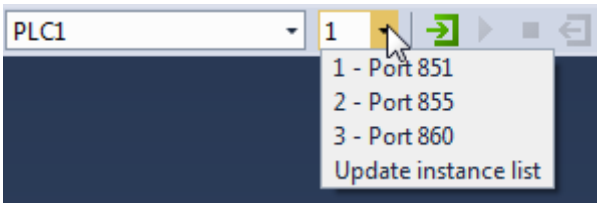
Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

This mechanism can be used to instance a PLC project, which results in one project instance per PLC runtime system. A separate ADS port number can be assigned for each project instance (see Command Change ADS port [▶ 236]). Furthermore, each project instance has its own links within a process image.



Each project instance has its own online mode, i.e. each project instance can be logged in and started individually, for example.



5.20.9 Command Save <PLC project name> as archive...

Function: The command opens the standard dialog for saving a file as an archive. The PLC project can be stored as a ***.tpzip** archive under the desired path.

Call: **File** menu, context menu

Requirement: The TwinCAT PLC project (<PLC project name>) is selected in the **Solution Explorer**.

The files and folders in the archive folder depend on the PLC project settings.

Content of *.tpzip	The *.tpzip archive folder contains the PLC project to be archived.
Command for opening	A tpzip archive can be reopened with the following command: Command Add Existing Item (Project) [▶ 86]
Note on PLC projects	The files and folders stored in the archive folder for the PLC project depend on the PLC project settings of this PLC project. Settings tab [▶ 112]

5.20.10 Command Send by E-Mail...

Symbol: 

Function: The command starts the email program that is set in the system and opens a new email with the archive file of the selected project as an attachment.

Call: File menu, context menu

5.20.11 Command Update project with target system…

Function: This command enables the selected PLC project to be updated to the project status of the connected target system.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected in the Solution Explorer and the sources of a PLC project are located on the connected target system.

5.20.12 Command Independent project file

Function: The PLC project is explicitly saved in a separate (*.XTI) file. This way, for example, the links of the PLC project are no longer saved in the TwinCAT project file, but in the XTI file of the PLC project.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

5.20.13 Command Deactivate

Function: The PLC project is deactivated and is ignored, for example, when building the TwinCAT project.

Call: Context menu of the PLC project in the **Solution Explorer**

Requirement: The PLC project is selected.

5.21 Context menu PLC project object (<PLC project name> Project)

5.21.1 Command Login

Symbol: 

Function: This command connects the programming system (the selected PLC project) with the target system (controller) and thus establishes online operation. An instance of the PLC project is created on the target system and loaded.

Call: **PLC** menu or **TwinCAT PLC toolbar options** or context menu of the PLC project object (<PLC project name>Project) in the **Solution Explorer**

Requirement: The PLC project is error-free and the target system is in Run mode.

Possible login situations:

- The PLC project does not yet exist on the controller: You will be prompted to confirm the download.
- The PLC project is already on the controller and has not been changed since the last download. Logging in takes place without further interaction with you.
- The PLC project is already on the controller, but has been changed since the last download. You will be prompted to choose one of the following options:
  - Login with Online Change (please refer to the notes in section "Command Online Change [▶ 138]")
  - Login with download
  - Login without changes

    At this point you can also update the boot project on the controller.
- An unknown version of the PLC project is already on the controller. You will be asked whether TwinCAT should replace it.
- A version of the PLC project is already running on the controller. You will be asked whether TwinCAT should nevertheless log in and overwrite the PLC program that is currently running.
- The PLC program on the controller is currently stopped at a breakpoint. You have logged out and changed the program: TwinCAT warns you that, in the event of an online change or download, the PLC will be completely stopped. This happens even if there are several tasks and only one of them is affected by the breakpoint.

Compiling the project before logging in

If a PLC project has not been compiled since its last change, TwinCAT compiles the project before logging in. This operation corresponds to the command **Compile in logged-out state**.

If errors occur during the compilation, a message dialog appears. The errors are displayed in the **Error List** view. You can then decide whether you want to log in without loading the program onto the controller.

See also:

- Command Build PLC project [▶ 242]

Error on login

If an error occurs while logging on to the controller, TwinCAT interrupts the loading process with an error message. The error dialog allows you to display the error details. If an exception error has occurred and the text *SOURCEPOSITION* is contained in the log message, you can use the command Display in Editor to display the relevant function in the editor. The cursor jumps to the line causing the error.

Output of information on the loading process

If TwinCAT loads the project onto the controller when logging in, the following information is displayed in the message window:

- Generated code size
- Size of global data
- Resulting memory requirement on the controller
- A list of the affected function blocks (for online change)

In online mode, you cannot change the settings of devices or modules. To change device parameters, you must log the PLC project out. Depending on the bus system, there may, however, be some special parameters that you can change in online mode.

TwinCAT stores the view configuration separately for online and offline mode. In addition, views that cannot be used in an operation mode are closed. For this reason, the view may change automatically when you log in.

5.21.2 Command Build PLC project

Symbol: 

Function: This command starts the compilation process or the code generation for the currently active PLC project.

Call: **Build** menu if a PLC project is currently selected, or context menu of the PLC project object (<PLC project name> Project) in the **Solution Explorer**

Requirement: The PLC project is selected.

During the compilation, TwinCAT carries out a syntactic test of all objects used in the PLC project. The compilation procedure is always carried out automatically when you wish to log the project in with a changed program. After the check has been completed, TwinCAT displays any error messages or warnings in the Error List [▶ 83] view.

Apart from that, the compilation information of the PLC project is created when building the project and saved in a local file (*.compileinfo) in the Solution.

If the program was not changed since the last error-free compilation process, it is not recompiled. If the syntactic test is to be performed anyway, use the Command Rebuild a PLC project [▶ 242].

5.21.3 Command Rebuild a PLC project

Function: The command starts the compilation process or the code generation for the currently active PLC project, even if it was last compiled without error.

Call: **Build** menu if a PLC project is currently selected, or context menu of the PLC project object (<PLC project name> Project) in the **Solution Explorer**

Requirement: The PLC project is selected.

When rebuilding the project, the project will first be cleaned (see also: Command Clean PLC project [▶ 243] and subsequently built (see also: Command Build PLC project [▶ 242]).

5.21.4 Command Check all objects

Function: The command initiates a compilation run, i.e. a syntax check for all objects located in the project tree of the PLC project. This is primarily useful when creating libraries or when processing library projects.

Call: Context menu of the PLC project object (<PLC project name> Project) in the **Solution Explorer**

As opposed to the Command Build PLC project [▶ 242], in which only the objects used are checked, when this command is executed the syntax of all objects in the PLC project is checked.

The command does not lead to code generation. No file with information on the compilation run is created in the project directory.

5.21.5 Command Clean PLC project

Function: This command deletes the local compilation information for the currently active PLC project and updates the language model of all objects.

Call: **Build** menu if a PLC project is currently selected, or context menu of the PLC project object (<PLC project name> Project) in the **Solution Explorer**

Requirement: The PLC project is selected.

The compilation information was created during the last building of the project or the last online change or download of the PLC project and was saved in a local file (*.compileinfo) in the project folder. Unless configured otherwise (see also: PLC project settings, Settings tab [▶ 112]), the compilation information is transmitted to the target system during an online change or download. When the PLC project is cleaned, only the local compilation information is removed. The compilation information on the target system is retained.

See also:

- Command Rebuild a PLC project [▶ 242]

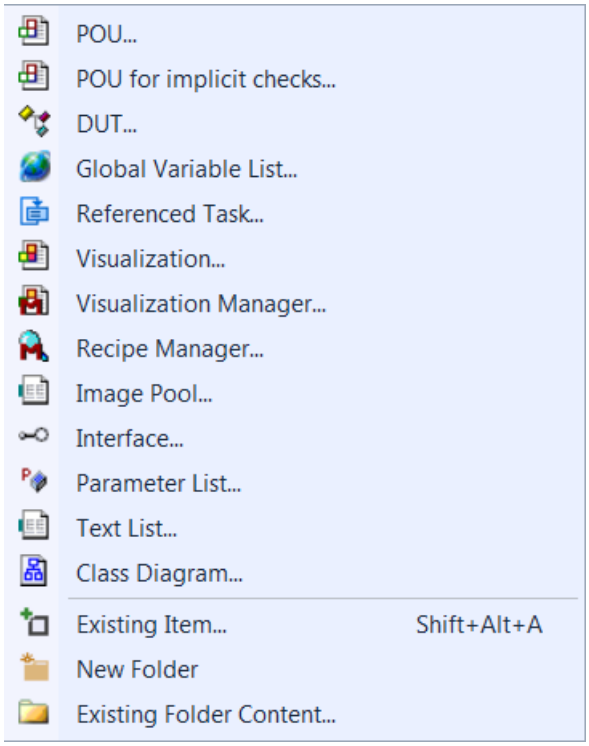
5.21.6 Add

5.21.6.1 Command Add New Item (object)

Function: The command opens a submenu with objects that contains all objects that can be inserted, depending on the current position in the PLC project tree.

Call: Context menu of the PLC project object (<PLC Project name> Project) or context menu of an already existing object or folder within the PLC project in the **Solution Explorer**

Requirement: If TwinCAT is to insert the object in the PLC project tree, select an existing object or a folder below which the new indented object is to be created.



See also:

- PLC documentation: Add objects

- PLC documentation: Creating and configuring a project

5.21.6.2 Command Add Existing Item (Object)

Symbol: 

Function: This command opens the standard browser dialog, via which you can search for a file and add it to the selected PLC project.

Call: Context menu PLC project object (<PLC Project name> Project) or context menu of an existing PLC folder in the **Solution Explorer**

Requirement: The PLC project object or the folder within a PLC project is selected in the TwinCAT project tree.

5.21.6.3 Command New folder

Symbol: 

Function: The command inserts a new folder in the PLC project.

Call: Context menu of the PLC project object (<PLC Project name> Project) or context menu of an already existing object or folder within the PLC project in the **Solution Explorer** > Add

Requirement: The PLC project object or the folder within a PLC project is selected in the TwinCAT project tree.

The command inserts the folder below the currently selected object in the project tree.

See also:

- PLC documentation: Adding objects

5.21.6.4 Command Add existing folder contents

Function: This command opens the standard browser dialog, via which you can search for and select a folder at file level. The contents of the folder are added to the selected PLC project.

Call: Context menu PLC project object (<PLC Project name> Project) or context menu of an existing PLC folder in the **Solution Explorer**

Requirement: The PLC project object or the folder within a PLC project is selected in the TwinCAT project tree.

5.21.7 Command Export to ZIP

Function: The command opens the standard dialog for saving the selected objects in ZIP format.

Call: Context menu of the PLC project object (<PLC Project name> Project) or context menu of an already existing folder within the PLC project in the **Solution Explorer**

Requirement: The PLC project object (<PLC project name>project) or the PLC objects are selected.

See also:

- Command Import from ZIP [▶ 245]
- PLC documentation: Exporting and importing a PLC project

5.21.8 Command Import from ZIP

Function: This command opens the standard dialog for importing objects from a ZIP file.

Call: Context menu of the PLC project object (<PLC Project name> Project) or context menu of an already existing folder within the PLC project in the **Solution Explorer**

Requirement: The PLC project object or the folder within a PLC project is selected in the TwinCAT project tree.

See also:

- Command Export to ZIP [▶ 244]
- PLC documentation: Exporting and importing a PLC project

5.21.9 Command Export PLCopenXML

Symbol: 

Function: The command opens a dialog for exporting objects of a project into an XML file in PLCopen format.

Call: Context menu of the PLC project object (<PLC Project name> Project) or context menu of an already existing folder within the PLC project in the **Solution Explorer**

Requirement: The PLC project object (<PLC project name>project) or the PLC objects are selected.

See also:

- Command Import PLCopenXML [▶ 245]
- PLC documentation: Exporting and importing a PLC project

5.21.10 Command Import PLCopenXML

Symbol: 

Function: The command opens a dialog for importing objects from an XML file in PLCopen format.

Call: Context menu of the PLC project object (<PLC Project name> Project) or context menu of an already existing folder within the PLC project in the **Solution Explorer**

Requirement: The PLC project object or the folder within a PLC project is selected in the TwinCAT project tree.

See also:

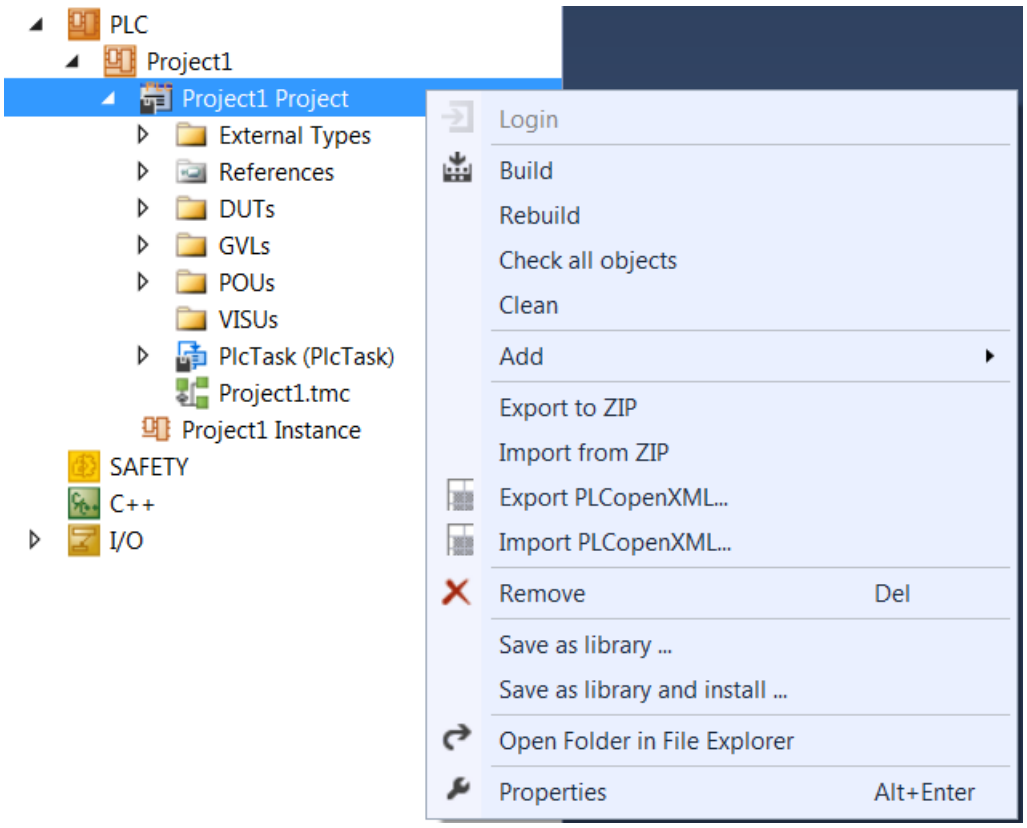
- Command Export PLCopenXML [▶ 245]
- PLC documentation: Exporting and importing a PLC project

5.21.11 Command Save as library

Function: The command opens the standard dialog for saving a PLC project as a PLC library.

Call: Context menu of the PLC project object (<PLC project name>project) in the Solution Explorer

A PLC project can be saved as a PLC library, in order to make source code available for other applications as a library and therefore via a defined interface. The command for saving a library is available in the context menu of the PLC project.

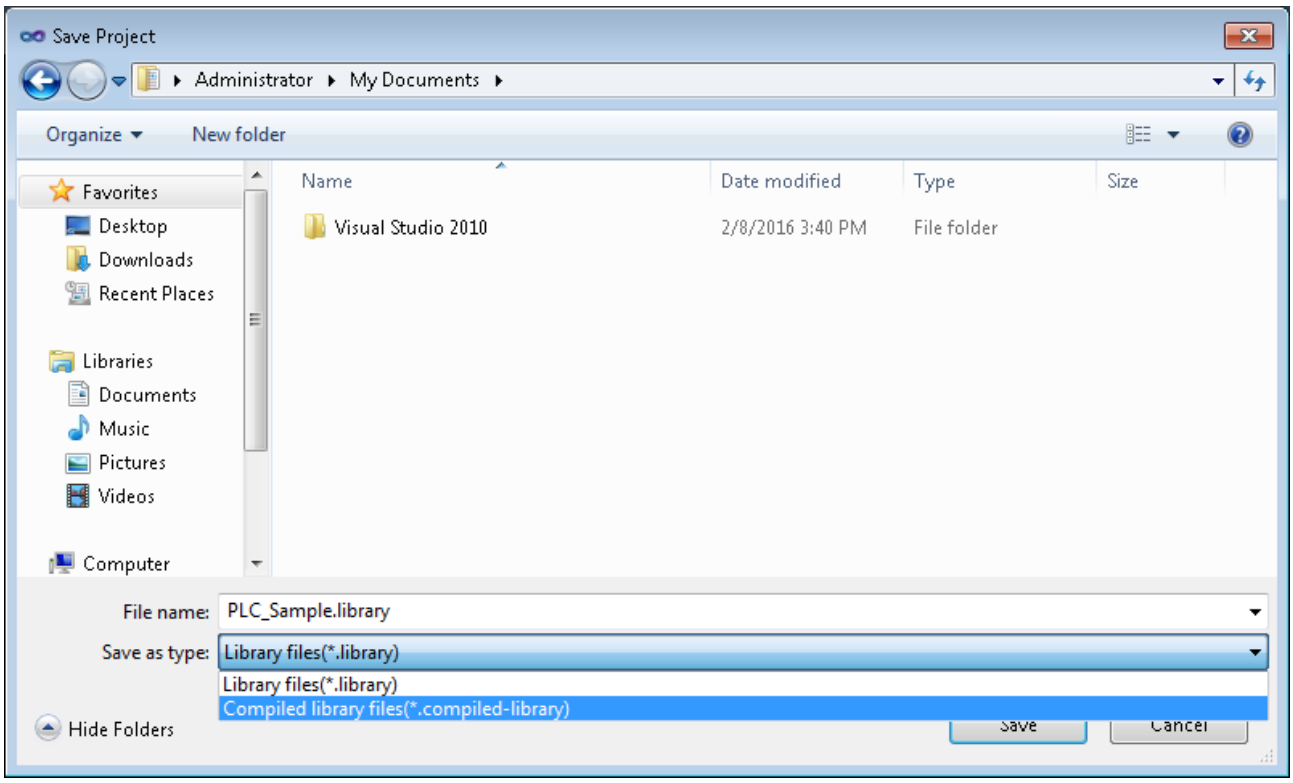


The command opens the standard dialog for saving a file in the file system. The existing project name is offered automatically; it can be modified by the user, if required. When a project is saved as a library, there is a choice between two library file formats:

- *.library (source library)
  - You can open a source library (for viewing and/or editing) by using the command **Add Existing Item**, which is available on the PLC node within the project tree.
  - You can "step" into a source library using the usual debug functionality.
- *.compiled-library (compiled library)
  - This file extension can be used to save a library project in a compiled format. An encrypted image of the precompile context of the library is stored, which means the implementations of the library function blocks are no longer accessible or visible.
  - A compiled library cannot be opened or debugged.
  - Otherwise, *.compiled-library files behave like *.library files. You can therefore install and reference them in the same way.
  - The source code of a library can be protected by using the compiled library format. In addition, the library files are smaller and the loading times are shorter.

Stepping is not possible

The usual debugging functions cannot be used for a *.compiled-library, i.e. "stepping" is not possible in a library function block of a *compiled-library.

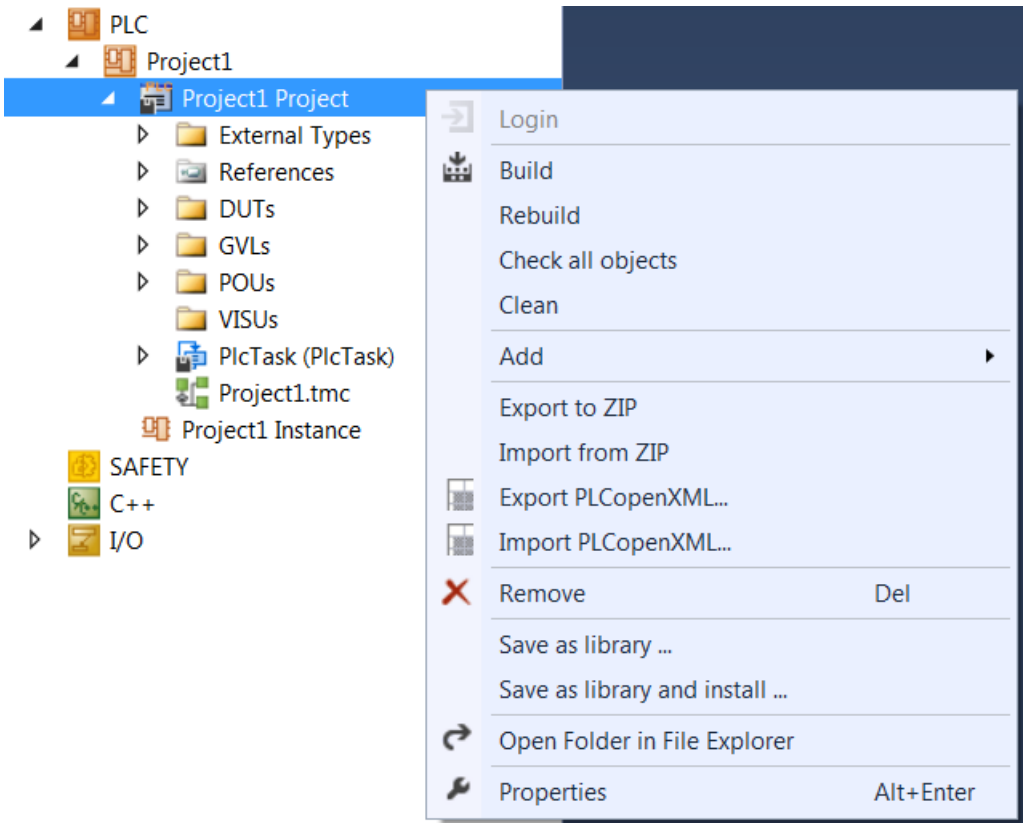


5.21.12 Command Save as library and install

Function: The command opens the standard dialog for saving a PLC project as a PLC library. In addition, the command installs the stored library in the library repository. The library can thus be inserted directly into a project via the Library Manager.

Call: Context menu of the PLC project object (<PLC project name>project) in the Solution Explorer

This command saves the PLC project as a PLC library and installs it in the Library Repository. The command for saving and installing a library is available in the context menu of the PLC project.



The installation of the library, which is performed in addition to the saving, is an extension of the Command Save as library [▶ 245], since the library is installed on the local system at the same time. The library is then immediately available for addition to a project via the Library Manager.

5.21.13 Command Properties (PLC project)

Symbol:

Function: This command opens an editor window in which the properties of the project and additional project-related information can be displayed and defined.

Call: Context menu of the PLC project object (<PLC project name> Project) or **Project** menu

Requirement: A project is open.

TwinCAT stores the PLC project properties directly in the PLC project.

Scope of PLC project properties

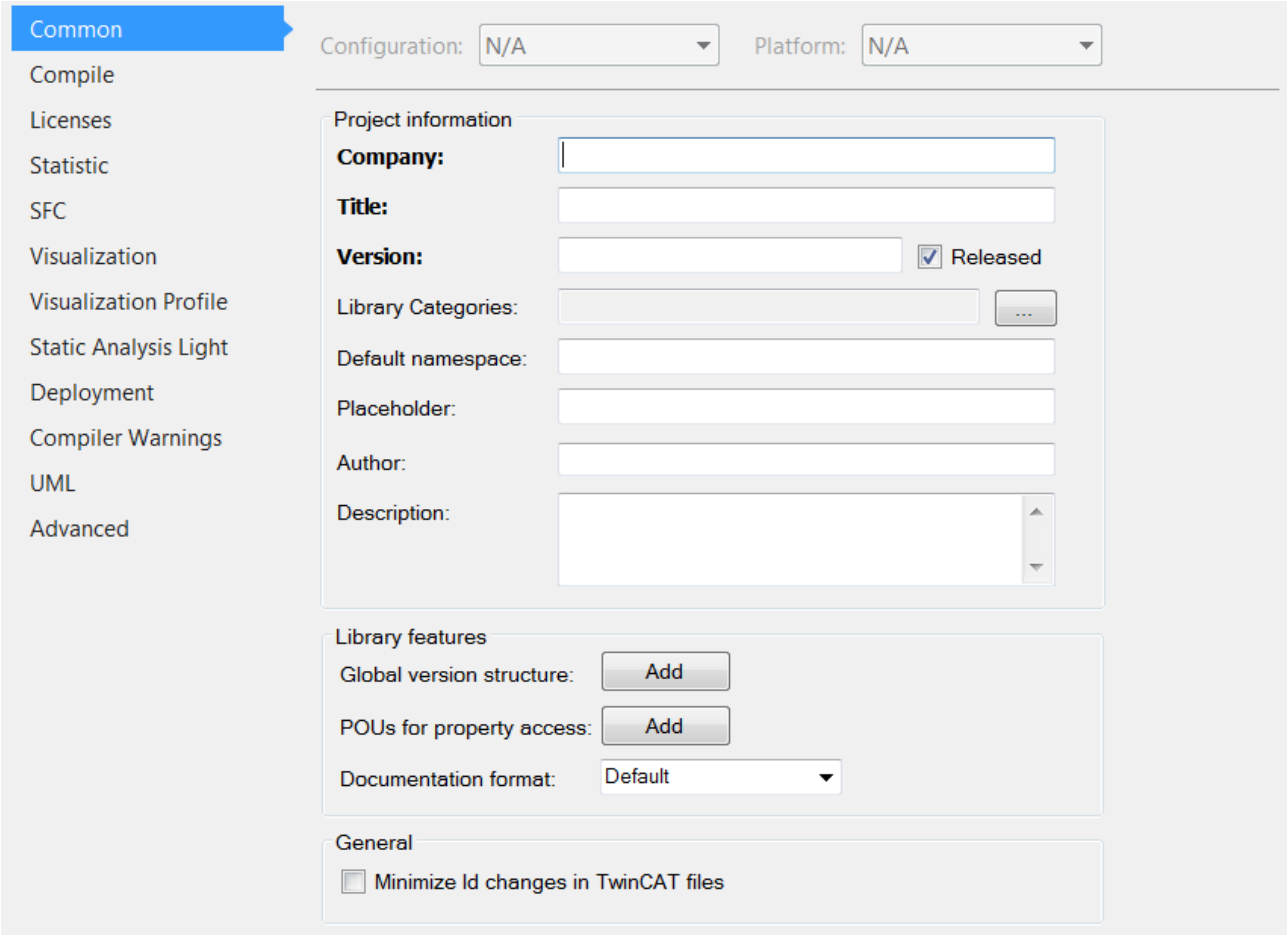
Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

See also:

- PLC documentation: Configuring a project

5.21.13.1 Category Common

The category **Common** contains general information and meta-information of the project file. TwinCAT uses this information to create keys in the **Properties** tab. For example, if the **Company** text field contains the name "Company_A", the **Properties** tab contains the **Company** key with the value "Company_A".



Project information

For a library project, a company, title and version must be entered in order to be able to install the library.	
Company	Name of the company, which created this project (application or library). In addition to the library category, it is used for sorting in the library repository
Title	Project title
Version	Project version, e.g. "0.0.0.1"
Released	 : Protection against modification enabled. Result: When you now edit the project, a prompt will appear asking you if you really want to change the project. If you answer this query once with Yes, the prompt will no longer appear when the project is edited again.
Library Categories:	Categories of the library project by which you can sort in the Library Repository dialog. If no category is specified, the library is assigned the category "Other". To assign it to another category, the category must be defined. Library categories are defined in one or several external description files in XML format. To assign the library, you can either call such a file or another library file that has already picked up information about the categories from a description file. Requirement: The project is a library project.
	The **Library Categories** dialog opens, in which you can add library categories.
Default namespace:	The default setting for the namespace of a library is the library title. Alternatively, a different namespace can be defined explicitly, either generally for the library at this point in the project information during library generation, or in the **Properties** dialog of the library reference for local use of the library in a project. The namespace of the library must be used as prefix of the identifier, in order to enable unambiguous access to a module that exists more than once in the project, or if the use of this prefix is enforced by the library property LanguageModelAttribute "qualified-access-only" ("Unambiguous access to library modules or variables"). If you do not define a standard namespace here, the name of the library file is automatically used as the namespace.
Placeholder	At this point, a default name for the placeholder can be specified, which represents or references this library. If a placeholder is not specified explicitly at this point, the default setting for the placeholder name of a library corresponds to the library title.
Author	Project author
Description	Brief description of the project (e.g. content, functionalities, general information such as "for internal use only", etc.)

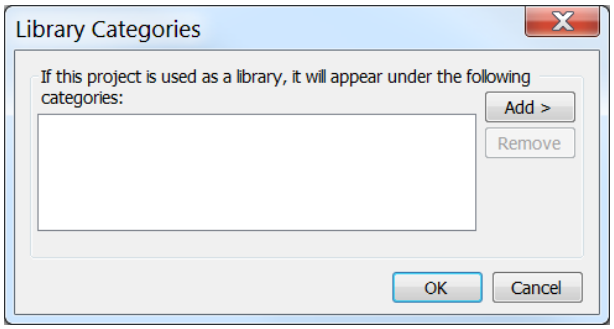
Library features

Creating a global version structure	Creates a global variable list in the PLC project, which contains the version information.
Automatically generate library information POU	**Add** button: POU objects of type "Function" are automatically created in the project tree, which can be used to access project properties in the application program. In this case, special functions are generated for the properties **Company**, **Title** and **Version** (F_GetCompany, F_GetTitle, F_GetVersion). If these functions have been added to the project by clicking **Add**, they can be removed from the project by clicking **Remove**.
Documentation format	Option **reStructuredText**: During library creation, comments that correspond to a specific format are restructured and displayed in the **Documentation** tab of the Library Manager in this customized view. This opens up additional options for library documentation.

General

Minimize ID changes in TwinCAT files	The GUIDs of the PLC objects (e.g. POUs) are linked to those of the PLC project (using XOR). This avoids changes to the GUIDs of the PLC objects when they are used several times in different projects.

Library Categories dialog



List of categories	List of categories assigned to the library project. They can come from several sources. When you have entered all the desired categories, confirm the dialog with **OK**.
Add	The commands **From Description File...** and **From Other Library...** appear.
Remove	TwinCAT removes the selected category.
From description file...	The **Select Description File** dialog appears, in which you can select a description file with the extension *.libcat.xml. The file contains command categories. When you exit the dialog with **Open**, TwinCAT applies these categories.
From another library....	The **Select Library** dialog appears, in which you can select a library (*.library) whose command categories are to be adopted. When you exit the dialog with **Open**, TwinCAT applies these categories.
OK	TwinCAT provides the categories as project information and displays them in the **Library Categories** field.
Cancel	Closes the dialog. The process is aborted.

See also:

- PLC documentation: Configuring a project
- PLC documentation: Using libraries

5.21.13.2 Category Compile

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring. Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **Compile** is used to configure the compiler options.

Settings

Compiler definitions	Here you can enter compiler definitions/"defines" (see {define} statements) and conditions for compiling the application (conditional compilation). A description of the available conditional pragmas can be found in section Conditional pragmas. The expression expr used in such pragmas can also be entered here. Several entries are possible in the form of a comma-separated list.
System compiler definitions	Available from TC3.1 Build 4024 The compiler definitions that have been set at System Manager level in the PLC project settings under Compiler definitions [▶ 111] are automatically adopted here.
Download application info	Available from TC3.1 Build 4024 Situation: You are loading a PLC project onto the controller that differs from the project already located there. In this case, a message window appears containing a **Details** button. Use this button to open the **Application information** window, which allows you to check the differences between the current PLC project and the PLC project on the controller. This involves comparing the number of function blocks, the data and the storage locations. The **Application information** window contains a brief description of the differences, for example: • Declaration of MAIN changed • Variable fbMyNewInstance inserted in MAIN • Number of methods/actions of FB_Sample changed  (Default): If this setting is enabled, the information on the contents of the PLC project is loaded onto the PLC. This enables an extended check of the differences between the current PLC project and the PLC project on the controller. The difference in the extended check option is that the **Application information** window contains an additional **Online comparison** tab, which shows a tree comparison view. This will tell you which POUs have been changed, deleted or added. The additional tab appears when you execute the blue underlined command in the lower area of the **Application information** window ("Application not current. Generate code now to display the online comparison?“).

Generate tpy file	Available from TC3.1 Build 4024 The tpy file contains, among other things, project, routing, compiler and target system information. It is the format used for describing a TwinCAT 2 PLC project. To ensure compatibility with existing applications, this file can be created for a TwinCAT 3 PLC project, if required. ☐ (Default): When the PLC project is created, no tpy file associated with the project is created. ☑: When the PLC project is created, a tpy file associated with the project is created and stored in the project folder. Please note that the value and configuration availability of this option depends on whether or not the TPY file is configured as a target file (see Settings tab [▶ 112]). • If the TPY file is enabled as a target file, the following happens: ◦ TwinCAT remembers the current status of the "Generate tpy file" option (= "original value", see below.). ◦ If this is not already the case, the option "Generate tpy file" is automatically activated next time the project is created. ◦ In addition, the "Generate tpy file" option is grayed out so that it cannot be disabled by the user as long as the TPY file is configured as a target file. • If the TPY file is subsequently disabled as a target file, the following happens: ◦ Next time the project is created, the "Generate tpy file" option is assigned its "original value" (see above.). ◦ In addition, the option is no longer grayed out, making it available again for configuration by the user.

Solution options

Compiler Version	Defines the compiler version that TwinCAT uses during compilation and during loading for compilation. Please note that this setting is not a substitute for the Remote Manager. If the PLC project is an application project, the Remote Manager should always be used for handling different Engineering versions. In this case, the compiler version should always be set to "latest". The compiler version setting is only relevant if the PLC project to be version-managed is a library project. It is advisable to save the oldest version of the library that will ultimately be used in practice. To this end, the compiler version must be set to the corresponding fixed version (e.g. "3.1.4018.0").
Maximum number of warnings	Refers to the maximum number of warnings that TwinCAT issues in the **Error List** view. The selection of displayed compiler warnings is defined in the category **Compiler warnings** in the **Project Settings** dialog.
Replace constants	☑: TwinCAT loads the value directly for each constant of scalar type, i.e. not for STRING, ARRAY or structures. In online mode, TwinCAT identifies the constants in the declaration editor or monitoring window with a symbol preceding the value. In this case, access via an ADR operator, forcing or writing, for example, is not possible. ☐ (Default): Access to constants is possible. The computing time increases slightly.

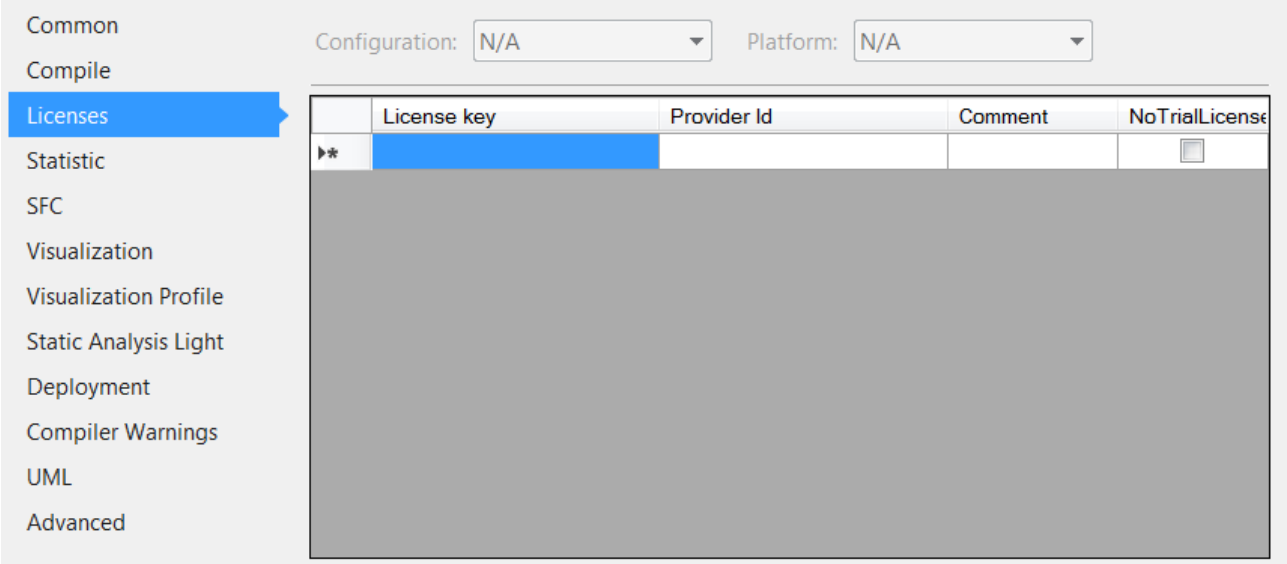
See also:

- Category Compiler Warnings [▶ 260]

5.21.13.3 Licenses category

In future, the **Licenses** category is intended to facilitate allocation of a TwinCAT 3 OEM license to a custom or proprietary library. This feature is not yet implemented.

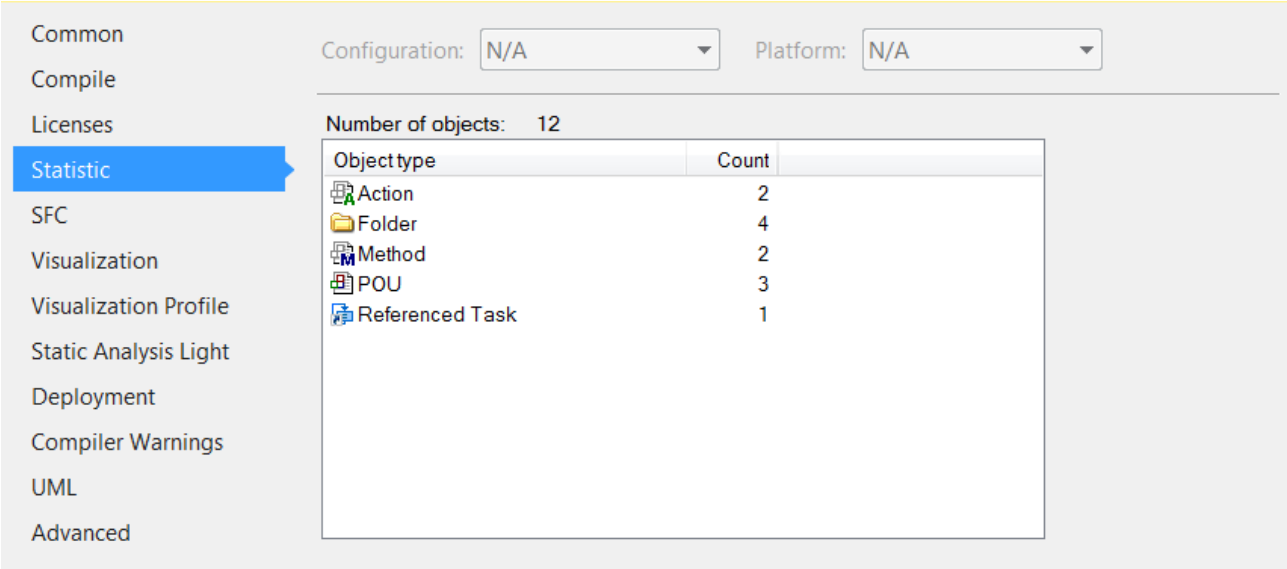
This category is therefore not yet supported in the current TwinCAT version (reserved for future use).



The user must currently query an OEM license for his own library in the code of the library. See Query of an OEM license in the PLC application.

5.21.13.4 Category Statistic

The category **Statistic** provides statistical information on how many objects of the different types are present in the project.



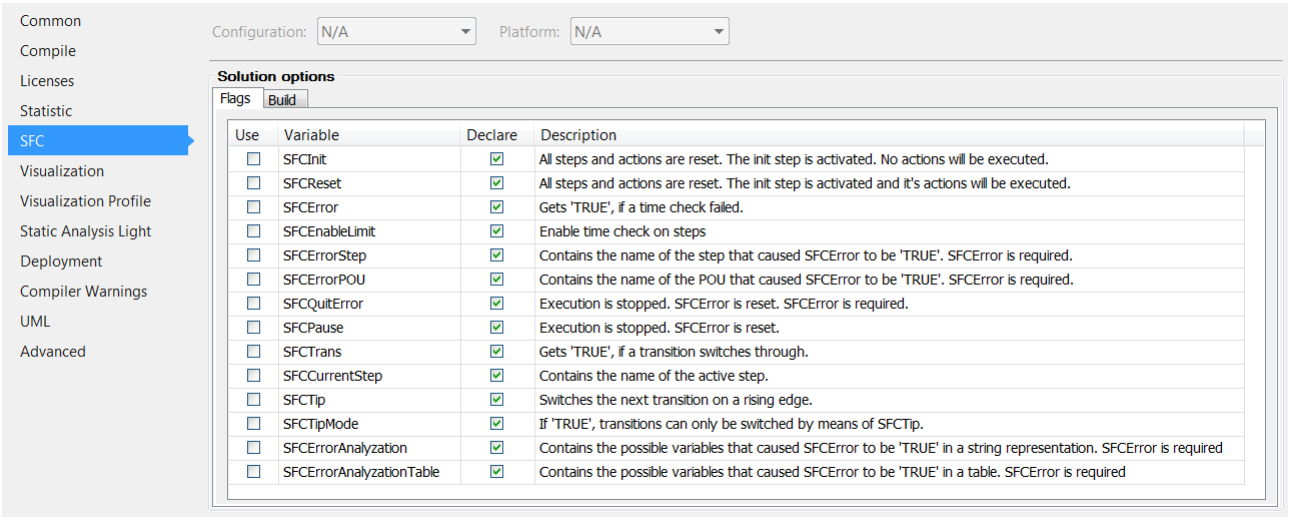
5.21.13.5 Category SFC

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **SFC** is used to configure the settings for SFC objects. Each new SFC object automatically has the configured settings in its properties.

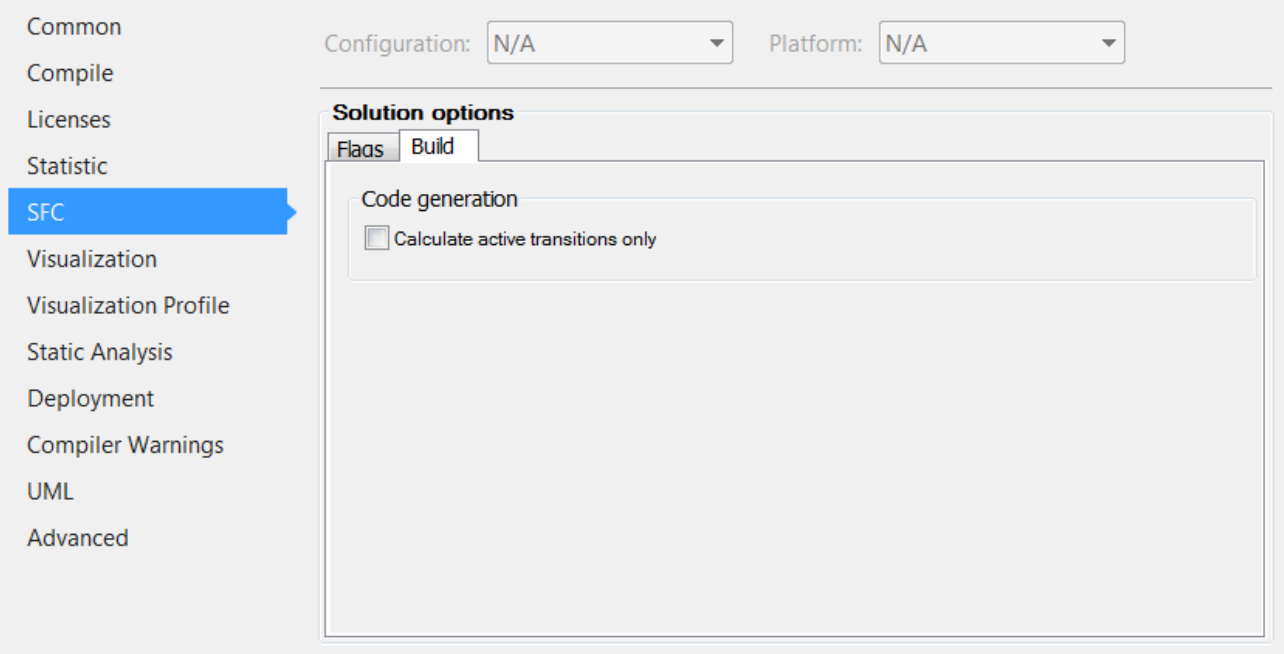
Flags tab



Implicitly generated variables (flags) for controlling and monitoring the processing in an SFC diagram.	
Use	: The corresponding variable is used.
Declare	: The corresponding variable is created automatically. Otherwise, if the usage is intended (Use is set), the user has to declare the variable.

An automatically declared flag variable appears in the declaration part of the SFC editor, but only in online mode.

Build tab



Code generation

Calculate active transitions only	: TwinCAT generates code only for transitions that are currently active.

See also:

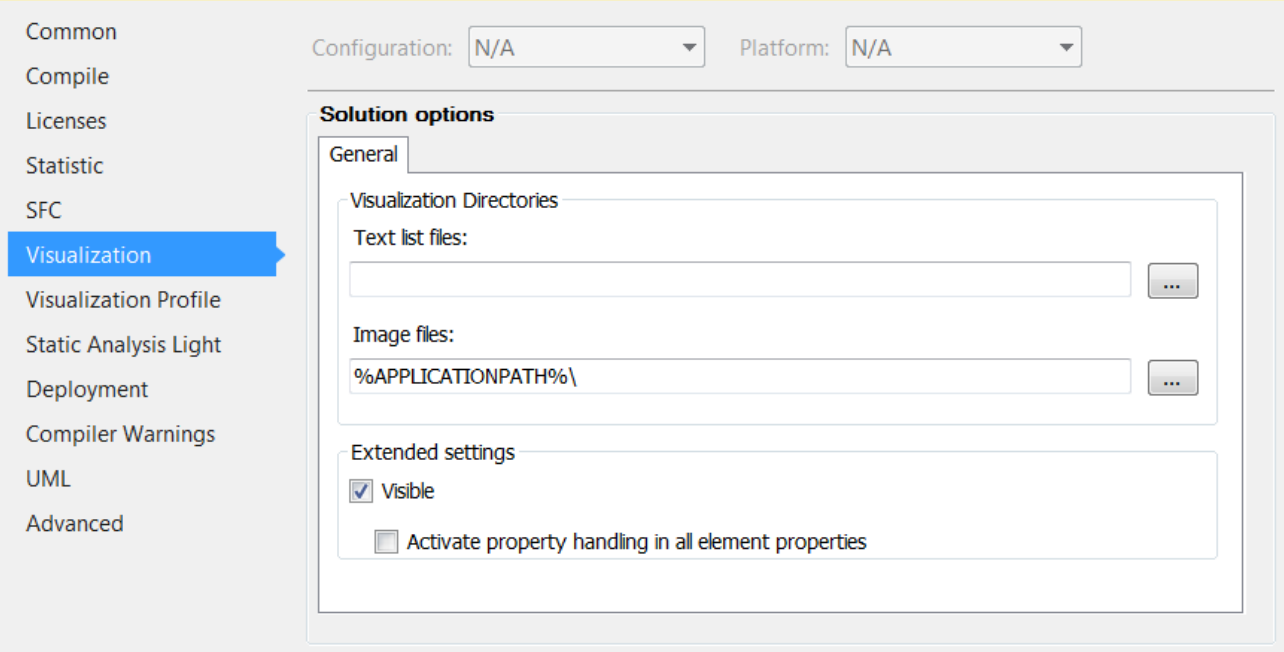
- SFC Flags

5.21.13.6 Category Visualization

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring. Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **Visualization** is used to configure the project-wide settings for objects of type Visualization.



General tab

Visualization Directories

Text list files	Directory containing text lists available in the project for configuring texts for different languages. TwinCAT uses this directory when exporting or importing text lists, for example. Click ... to bring up the **Find Folder** dialog, which allows you to select a directory in the file system.
Image files	Directory containing image files available in the project. Multiple folders are separated by semicolons. TwinCAT uses this directory when exporting or importing image files, for example. Click ... to bring up the **Find Folder** dialog, which allows you to select a directory in the file system.

Extended settings

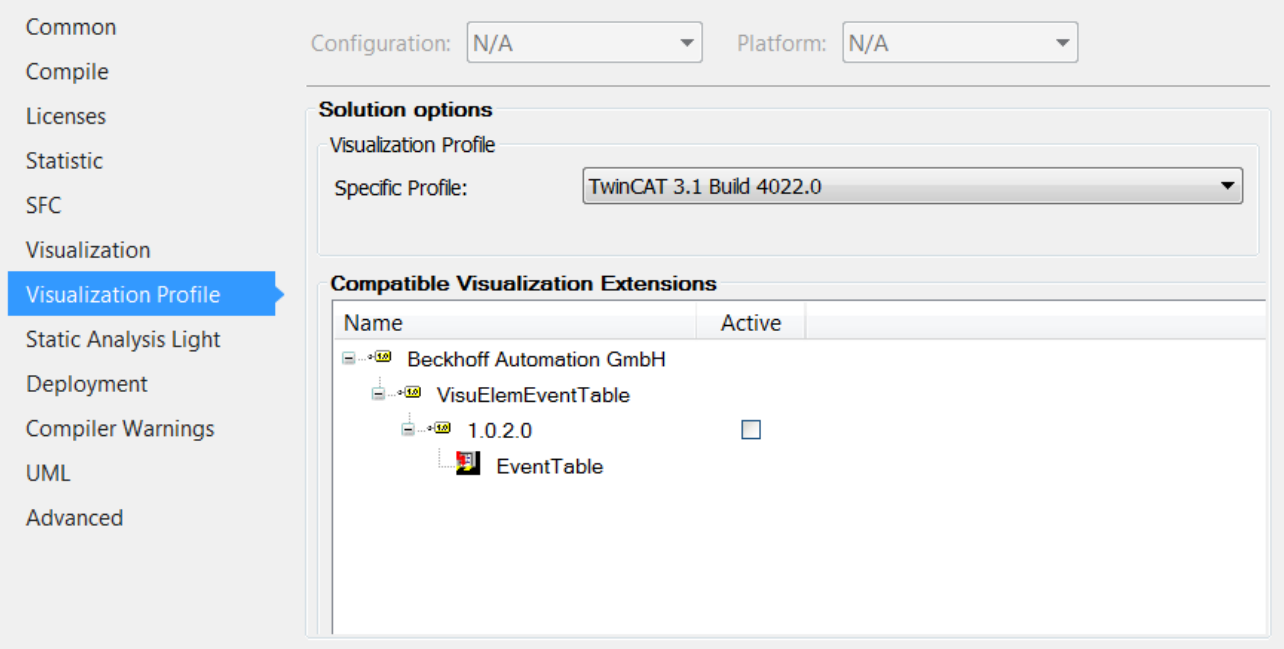
Enables property handling in all element properties	: You can also configure a visualization element in those of its properties, in which you select an IEC variable, with a property. TwinCAT then generates additional code for properties handling when compiling a visualization. Requirement: Your IEC code contains at least one object of type Interface property, i.e. a property . MAIN (PRG) Property_A Get Set Requirement: **Visible** is enabled.

5.21.13.7 Category Visualization Profile

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The **Visualization Profile** category allows you to set the visualization profile.



Visualization profile

Specific Profile	Profile, which TwinCAT uses in the project and which determines the visualization elements that are available in the project. The selection list contains all previously installed profiles.

5.21.13.8 Category Static analysis

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **Static analysis** defines the checks that are taken into account in the static code analysis.

Static Analysis Light:

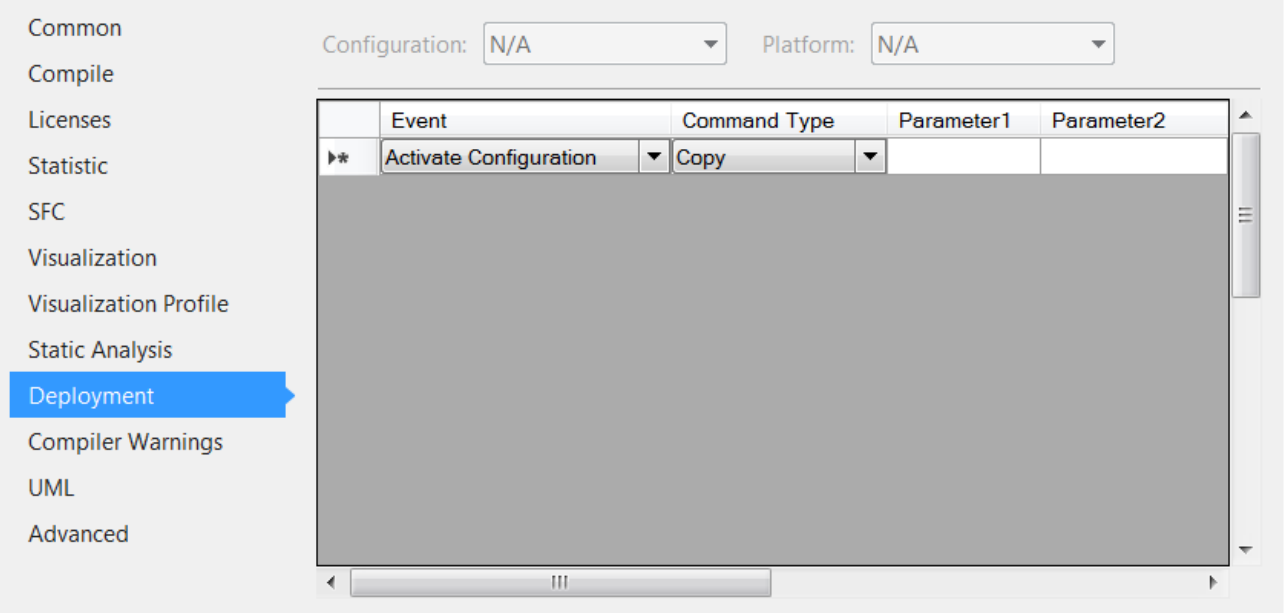
- If you have not activated the additional TE1200 Engineering license, you can use the license-free version of Static Analysis (Static Analysis Light), which contains some coding rules. The free Light version enables you to familiarize yourself with the basic handling of the product, for example, based on a reduced set of functions.
- For more information about Static Analysis Light see:
  PLC documentation: Programming a PLC Project > Checking syntax and analyze code > Code analysis (Static Analysis)

Static Analysis Full:

- If you have enabled the additional TE1200 Engineering license, the full range of Static Analysis functions is available (saving and loading settings, more than 100 coding rules, naming conventions, metrics, forbidden symbols).
- For more information about Static Analysis Full see: TE1200 Static Analysis.

5.21.13.9 Category Deployment

The category **Deployment** is used to set up commands that are to be executed during the installation and startup of an application.



The following events are available, after which the commands listed in the list can be called:

Activate Configuration	The required command is called up after the configuration has been enabled.
Plc Download	The required command is called up after the PLC application has been downloaded to the target system.
Plc Online Change	The required command is called after a successful online change.
Plc After Compile	The required command is called after a compilation of the PLC application.

The following commands can be executed:

Copy	Copies files from parameter 1 (source path) to a location specified in parameter 2 (target path).
Execute	Executes the application or script listed under parameter 1.

Source and target paths can contain virtual environment variables, which TwinCAT resolves accordingly.

The following environment variables are supported:

Virtual environment variable	Registry value	Default value
%TC_INSTALLPATH%	InstallDir	C:\TwinCAT\3.**x** \
%TC_TARGETPATH%	TargetDir	C:\TwinCAT\3.**x** \Target\
%TC_BOOTPRJPATH%	BootDir	C:\TwinCAT\3.**x** \Boot\
%TC_RESOURCEPATH%	ResourceDir	C:\TwinCAT\3.**x** \Target\Resource\
%SOLUTIONPATH%	-	Location of the solution file

Registry values are stored in the registry under the following key: *\HKLM\Software\Beckhoff\TwinCAT3.*

Example:

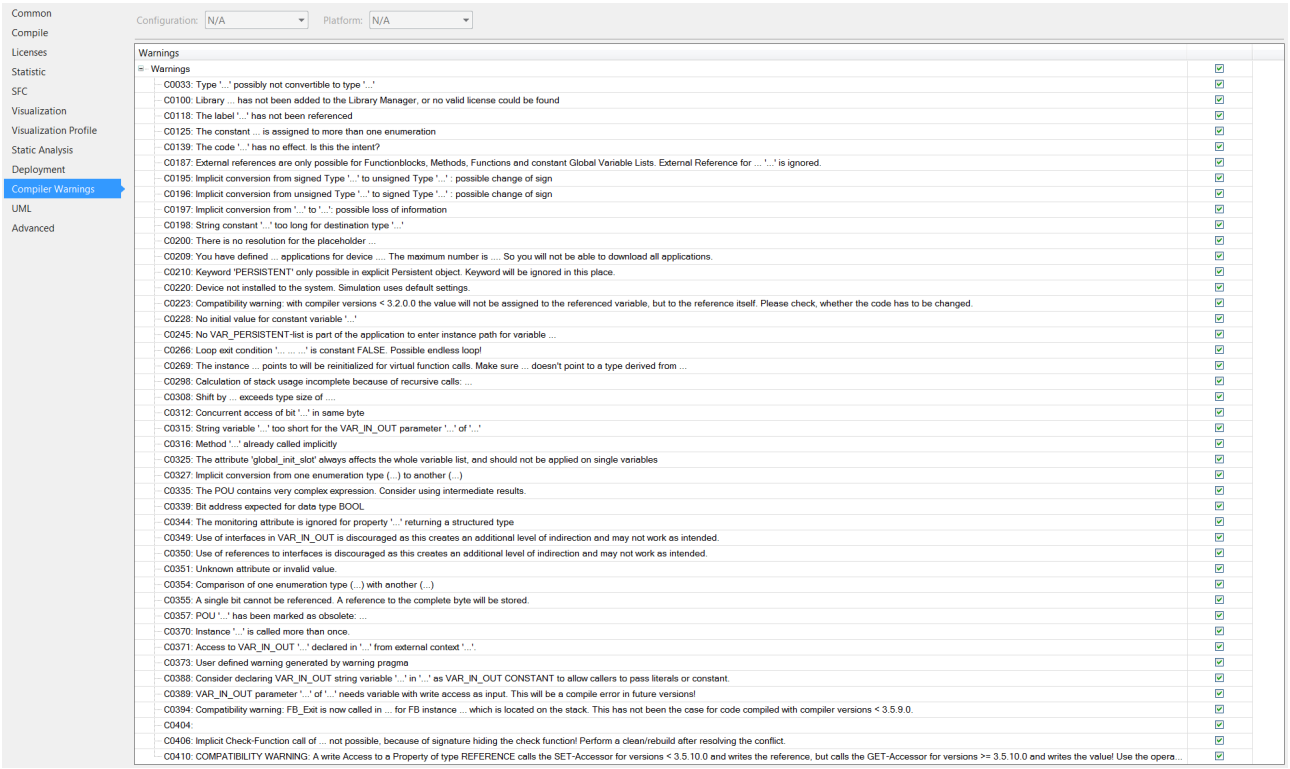
In the following example, the file *SampleFile.xml* file is copied from the Config project subfolder of the solution to the folder *C:\plc\config* on the target system.

Event	Command Type	Parameter1	Parameter2
Activate Configuration	Copy	%SOLUTIONPATH%Config\SampleFile.xml	C:\plc\Config\SampleFile.xml

5.21.13.10 Category Compiler Warnings

The **Compiler Warnings** category is used to select the compiler warnings that TwinCAT displays in the message window during a compilation run.

You can specify the maximum number of listed warnings in the **Compile** category.



See also:

- Command Build PLC project [▶ 242]
- Category Compile [▶ 251]

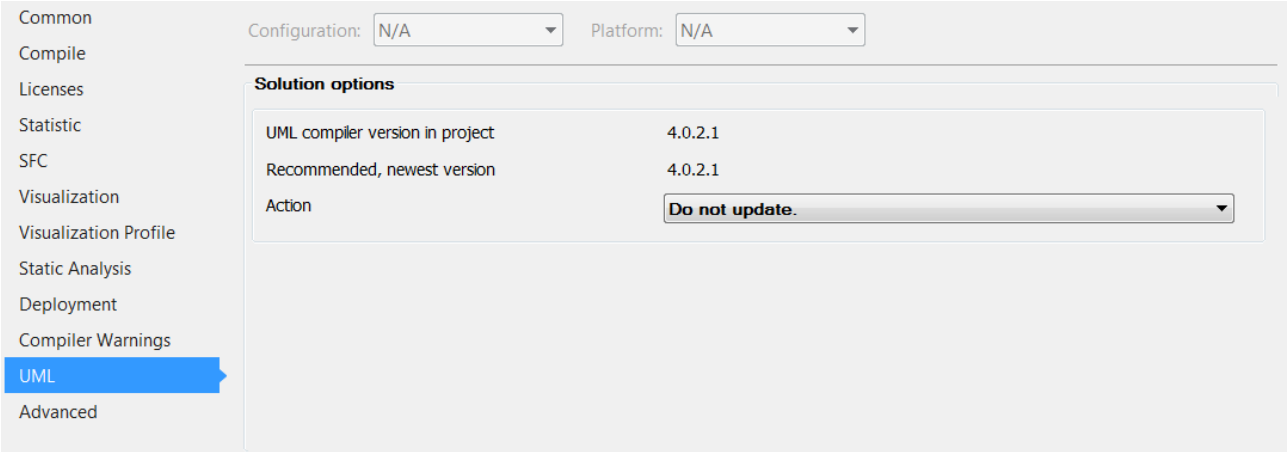
5.21.13.11 Category UML

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

In the category **UML** you can change the UML compiler version. This setting is only relevant when using the UML Statechart.

For more information on the configuration options, please refer to section "UML Compiler Version" of the TF1910 TC3 UML documentation.



5.21.13.12 Category Advanced

Scope of PLC project properties

Note that the scope differs between different project properties!
Some properties affect only the PLC project whose properties you are currently configuring.
Other properties, on the other hand, affect all PLC projects in the development environment. Such properties, which you can change in the project properties of a PLC project, but which also affect all other PLC projects, are titled **Solution options**.

The category **Advanced** is used to configure advanced properties.

Write options

Engineering incompatibility of file version 1.2.0.0 (or higher) with TC3.1 < build 4024

Please note that objects saved with file version 1.2.0.0 (or higher) cannot be loaded with Engineering versions lower than TC3.1.4024!

Since an object is automatically saved with file version 1.2.0.0 when using the optional Base64 format, objects with Base64 format cannot be loaded with Engineering versions lower than TC3.1.4024.

If a PLC project contains objects with the file version 1.1.0.1 and objects with the file version 1.2.0.0, the 1.1.0.1 objects are loaded with an Engineering version lower than TC3.1.4024. Objects with file version 1.2.0.0 are not loaded.

The file version of a file saved with file version 1.2.0.0 can be reset to 1.1.0.1 using XAE version TC3.1.4024 or higher.

Write object content as („Write object content as“)	Available from TC3.1 Build 4024 **Background information:** From build 4024, **Base64** introduces a new memory format that is optionally available for the following PLC objects: • POUs where the POU body is programmed in a graphical implementation language ◦ Sequential Function Chart (SFC) ◦ FBD/LD/IL (Function Block Diagram/Ladder Diagram/Instruction List) ◦ CFC (Continuous Function Chart and page-oriented CFC) ◦ UML class diagram and state diagram • POUs with a subelement (e.g., action, method) that is programmed in a graphical implementation language (for graphical languages see first key point) • Visualizations • Visualization Manager • Text lists • Recipe manager • Image pool Up to now, these objects were saved as XML by default. From build 4024 you can configure whether these object types should be saved as XML or Base64. **Advantages of Base64 over XML:** Base64 results in compressed storage, compared to XML. As a result, improved performance can be achieved with file access to these objects, which can be used, for example, when loading, moving or copying objects. **Setting option for the standard storage format:** For a PLC project, the setting "Write object content as" in the PLC project properties can be used to define the standard storage format for the object types mentioned above. The selected standard storage format is only used with newly added objects (exception: not with newly added POU sub-objects. Example: A POU is saved as an XML and the standard storage format is configured as Base64. If a graphic sub-object is then added to the POU, the storage format of the POU and thus of the sub-object as XML remains unchanged). The storage format of an existing object with a non-standard storage format is not automatically changed when the object is changed and saved. The storage format of an existing object can be changed individually via the Properties window (see below). Alternatively, when changing the standard storage format, there is the option of adopting the newly selected storage format for all existing objects. If you change the storage format at this point, a corresponding query window appears. The following options are available for the setting "Write object content as": • **XML** (default): The PLC objects mentioned above are saved in XML format by default. ◦ Objects with this storage format are stored in file version 1.1.0.1. • **Base64:** The PLC objects mentioned above are saved in Base64 format by default. ◦ Objects with this storage format are stored in file version 1.2.0.0. Please note that objects with the file version 1.2.0.0 (or higher) cannot be loaded with Engineering versions < TC3.1.4024! **Individual setting option of the storage format:** The storage format of an object can be configured individually for the object types mentioned above in the **Properties** window of the object. For more information see the description of the properties [▶ 90] (**Format** property).

Write product version in files („Write product version in files“)	Available from TC3.1 Build 4024 The product version indicates which plug-in version was used to save a PLC file (e.g., a function block). The setting of this checkbox is valid for the whole project and is the default setting for all modified or newly added PLC objects located in this PLC project.  (Default): The product or plugin version is written into the file (the version is not visible in XAE; it shows up if the file is analyzed at file level). • If you change the setting from disabled to enabled, a query window appears in which you can select whether to add the product version to all existing files. • Use case for the enabled option: This setting can be used to include the file version in the file for debugging or tracking purposes, for example. • Please note the following: If the file is saved with a different product version, this leads to a change of this file, which shows up as a file difference when using source code management systems.  : The product or plugin version is not written to the file. • If you change the setting from enabled to disabled, a query window appears in which you can select whether to remove the product version from all existing files. • Use case for the disabled option: This setting can be used if the product version is not of interest. This minimizes changes to files with regard to source code management systems.
Write object content with profile: (“Write object content with Profile”)	The profile defines the format in which objects are saved. With Build 4024, for example, new functionalities were added for the PLC HMI. For this reason, visualization files saved with Build 4024 cannot be directly opened with older builds. If you set a 4022 profile here, then the visualization files will be saved in the appropriate format and can be opened with Build 4022. **Requirement:** So that, for example, the 4022 profile is available in the drop-down menu, either a 4022 Remote Manager installation must be carried out or the current 4024 XAE installation must have been installed via a previously existing 4022 XAE installation.

Multiuser options

Use Multiuser („Use Multiuser“)	Available from TC3.1 Build 4024  (Default): The multiuser functionality of the PLC project is not enabled.  : The multiuser functionality of the PLC project is enabled. Please also refer to the further information in the multiuser documentation.

Solution options

Secure Online Mode („Secure Online Mode“)	☐ (Default): For security reasons, the user is always prompted to confirm the execution of the following commands when they are called. • Activate configuration • Restart TwinCAT System in Config/Run Mode • Reset cold • Reset origin ☑ : In addition to the above commands, for which a confirmation prompt appears by default, the following commands will also prompt you to confirm. • Start • Stop • Single Cycle
Autoupdate Visu Profile	This option allows you to configure the automatic update behavior of the visualization profile. When you open a PLC project that uses an outdated visualization profile, a warning appears in the message window ("New Version found for Visualization profile"). ☑ : In such a case, the visualization profile version is automatically set to the latest version if the option **Autoupdate Visu Profile** is enabled. With such an automatic update of the visualization profile version, a corresponding warning is displayed in the message window (e.g., "Visualization profile set from' TwinCAT 3.1 Build 4020.10' to' TwinCAT 3.1 Build 4022.0'"). ☐ (Default) If the **Autoupdate Visu Profile** option is disabled, the visualization profile version is not changed automatically. By double-clicking on the warning "New Version found for Visualization profile", you can open the ProfileUpdate dialog in which you can manually change the visualization profile version.
Autoupdate UML Profile	This option allows you to configure the automatic update behavior of the UML compiler version. If you open a PLC project, in which an outdated UML compiler version is used, a corresponding warning appears in the message window ("new version for UML found"). ☑ : In such a case, the UML compiler version is automatically set to the latest version if the option **Autoupdate Uml Profile** is enabled. In the case of such an automatic update of the UML compiler version, a corresponding warning will be displayed in the message window (e.g., "UML set from '4.0.2.0' to '4.0.2.1'"). ☐ (Default): If the option **Autoupdate UML Profile** is disabled, the UML compiler version is not changed automatically. Double-click on the warning “new version for UML found” to open the ProfileUpdate dialog, in which you can change the UML compiler version manually. For more information, see UML Compiler Version.

More Information:
www.beckhoff.com/te1000

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com